QUANTICOL

A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours

quantiCOL

http://www.quanticol.eu

D3.3

Combining spatial verification with model reduction and relating local and global views

Revision: 1.0; March 30, 2017

Author(s): Mieke Massink (Ed.) (CNR), Maurice ter Beek (CNR), Luca Bortolussi (CNR), Vincenzo Ciancia (CNR), Stefania Gnesi (CNR), Jane Hillston (UEDIN), Diego Latella (CNR), Michele Loreti (IMT), Mirco Tribastone (IMT), Andrea Vandin (IMT)

Due date of deliverable: Month 48 (March 2017)

Actual submission date: March 30, 2017

Nature: R. Dissemination level: PU

Funding Scheme: Small or medium scale focused research project (STREP)
Topic: ICT-2011 9.10: FET-Proactive 'Fundamentals of Collective Adaptive Systems' (FOCAS)
Project number: 600708

Coordinator: Jane Hillston (UEDIN)
e-mail: Jane.Hillston@ed.ac.uk
Fax: +44 131 651 1426

Part. no.	Participant organisation name	Acronym	Country
1 (Coord.)	University of Edinburgh	UEDIN	UK
2	Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo"	CNR	Italy
3	Ludwig-Maximilians-Universität München	LMU	Germany
4	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland
5	IMT Lucca	IMT	Italy
6	University of Southampton	SOTON	UK
7	Institut National de Recherche en Informatique et en Automatique	INRIA	France

COOPERATION

Executive summary

This final Deliverable of Work Package 3 describes the main achievements obtained during the last reporting period for all three tasks of the work package (and in part during the second reporting period regarding Task 1.3) concerning the development of the theoretical foundations of novel, scalable and spatial formal analysis techniques and the underlying theories to support the design of large scale CAS. During the first two reporting periods of the project a number of innovative analysis techniques have been developed that are highly scalable. Some of these are based on mean field approximation techniques, others involve statistical model checking and machine learning techniques. For all these cases additional model reduction techniques have been developed to further improve scalability of analysis, for example to reduce the number of ordinary differential equations (ODEs) that need to be solved or the number of populations that need to be considered. For what concerns spatial verification several spatial and spatio-temporal logics have been developed for which efficient verification techniques have been created based on model checking and monitoring techniques. In particular, Spatial Logic for Closure Spaces (SLCS), based on the formal framework of closure spaces, and Spatial Signal Temporal Logic (SSTL) extending Signal Temporal Logic (STL) with some of the spatial operators from SLCS in a monitoring setting. Finally, suitable extensions of a software product line engineering (SPLE) approach for CAS were developed, among which family-based verification of behavioural aspects of CAS.

In the third and final reporting period all these techniques have been further extended and some combined, implemented and applied to the case studies of the project. Some of the main achievements are: the extension of the fluid model checking algorithms incorporating various kinds of rewards (or costs); study of the conditions under which continuous time population models can be analysed based on discrete time mean field model checking techniques; approximation of probabilistic reachability; development of a front-end language for FlyFast to deal with components and predicate-based interaction; extension of SLCS with temporal operators and with collective operators; combination of statistical and spatio-temporal model checking; application of an extended version of SLCS on Medical Imaging; combination of SSTL with machine learning; development of CTMC and ODE based behavioural equivalences for CAS and related minimisation algorithms; definition of an efficient family-based model checking procedure for SPLE models; development of a tool for quantitative analysis of probabilistic and dynamically reconfigurable SPLE models via statistical model checking; variability-aware software performance models.

All these developments are briefly described in the three main sections of this deliverable reflecting the three tasks of Work Package 3.

Contents

1 Introduction

This deliverable reports on results achieved in the final reporting period of all three tasks of Work Package 3. In particular Task 3.1 "Spatial Stochastic Logics and Scalable Verification", Task 3.2 "Abstraction Techniques for Scalability Beyond Population Size" and Task 3.3 "Relating Local and Global System Views with Variability Analysis". In addition, in line with the Description of Work, it reports on results of Task 3.1 obtained during the second reporting period that were not covered in earlier deliverables of the project.

A common objective of all three tasks was to make substantial contributions to the theoretical foundations of scalable and spatial formal analysis methods that could form the basis for the development of a formal verification framework for Collective Adaptive Systems (CAS). Mean field and fluid approximation techniques, in combination with successful formal verification techniques based on logic and process algebras, form a recurring and essential key element in the research of all the tasks of Work Package 3. They have been particularly useful to overcome the long standing state space explosion problem in formal verification techniques such as model checking. However, the research presented in this deliverable shows that this idea can be taken much further.

Where in the previous deliverables fluid model checking and on-the-fly mean field model checking have been developed for the analysis of *local* reachability properties of an individual object in the context of one or more large populations, in this deliverable it is shown how fluid model checking can be extended to address time-bounded *global* reachability properties of the system, for example, properties concerning the probability that the size of a particular population exceeds certain levels within a certain time interval. Moreover, it is shown how fluid model checking can be extended to deal with various types of *reward* properties.

The discrete time, probabilistic mean field model checking approach, that led to the open source model checking tool FlyFast, has been further improved and used for a number of case studies, such as the analysis of a benchmark gossip protocol, described in Deliverable 5.3, and a bike sharing model, described in Deliverable 4.3, where a CaSL model is reduced to a FlyFast model. The latter example is based on a bike sharing model that was also described in Deliverable 3.1. That example also illustrates that, under some suitable conditions, the discrete time approach can be used to approximate fluid model checking results. As a further step towards the integration of FlyFast with specification languages based on a predicate-based communication paradigm, a front-end for the FlyFast modelling language has been developed that incorporates components and predicate-based communication inspired by the CARMA language that was presented in Deliverable 4.2.

Fluid approximation techniques play also a key role in the development of model reduction. In particular the full characterisation of equivalence relations for Ordinary Differential Equations has been addressed. These relations were in part developed during the previous reporting periods, but have now been refined, and efficient, fully automatic minimisation techniques have been developed that can be used in a much more general setting. Such model reduction techniques, that go beyond fluid approximation, are important because they enable the analysis of models of a size that was impossible to handle with existing methods. Furthermore, these equivalences offer possibilities for symbolic computation. This leads, for example, to very efficient methods for the analysis of models with one or more parameter variables.

Mean field and fluid approximation techniques have also led to interesting contributions in the area of software product lines and variability analysis where symbolic solutions have been developed that allow for the efficient analysis of complete families of large scale systems at once.

However, mean field and fluid approximation techniques are not suitable for the analysis of *all* large scale CAS. This is for example the case in systems that involve uncertain parameter values, which occurs frequently in performance analysis. Exploiting a technique involving simulation-based model checking has shown to be a promising solution in such cases. This technique, also known as *smoothed model checking*, exploits properties of Gaussian processes to verify properties of models with several parameters.

In the context of Software Product Lines (SPL) and variability analysis, the large scale of the models is not always due to the presence of large populations, but rather to the large number of combinations of possible features that are present in families of products. To address such problems, a family-based approach to model checking has been pursued. In particular, an Eclipse-based tool for the quantitative feature-oriented language QFlan, introduced during the second reporting period, has been developed, using statistical model checking, to

address properties such as quality of service, reliability, or performance of dynamically reconfigurable product lines.

Finally, in the previous reporting periods several spatial and spatio-temporal logics have been proposed and efficient related model checking and monitoring algorithms have been developed. In the third reporting period this work has proceeded by developing novel spatial logic operators on one hand, and by the development of more efficient prototypes. Some of these have been made available as an Eclipse plug-in and their front-end made compatible with the CARMA language. Others have been developed as a stand-alone model checker and have been shown to be applicable not only to CAS but also to unforeseen areas such as medical imaging. Spatio-temporal model checking has also been combined with statistical model checking exploiting a feature of the MultiVeStA tool to analyse a property on all points in space simultaneously for each simulation run, making it feasible to apply stochastic spatio-temporal model checking on large CAS such as a bike sharing system of the size of that of London. Furthermore, such model checking and monitoring algorithms have been used to verify emergent spatio-temporal logic properties of systems modelled as reaction-diffusion systems as illustrated on a well-known example of Turing's work on morphogenesis.

We present the results in more detail in the following sections, one for each of the three tasks of the work package. Each section first briefly recalls the relevant previous results obtained and the objectives for the current reporting period after which the new results are presented organised by topic.

This deliverable reports on 40 publications (of which 2 submitted) of Work Package 3. Of these, 33 appeared during the final reporting period and 7 during the second reporting period. The latter are related to the second phase of Task 3.1 and were not yet described in previous deliverables of this work package. The publications are more or less evenly distributed over the three tasks. There were 15 joint publications with authors from two or more project partners.

The structure of the Deliverable follows the division of Work Package 3 in the three tasks. Section 2 presents an overview of the achievements in the context of Task 3.1 on spatial and scalable verification. Section 3 discusses the results on abstraction techniques and model reduction, and Section 4 presents results obtained on variability analysis for CAS. Section 5 presents the conclusions, listing the main achievements, the relation to other work packages of the project and provides a brief foresight on future work. Finally, three lists of references are presented: one list with the publications for Work Package 3 that appeared during the final reporting period; one list with publications related to Task 3.1 that appeared in the second reporting period and a list with earlier or related publications.

2 Scalable and Spatio-temporal Model Checking

2.1 Background

Scalable and spatio-temporal model checking has been one of the three main research themes of WP3 and has been developed in the context of Task 3.1 throughout the project duration. We briefly recall the main results from previous periods and provide an update on their finalisation and site of publication during the last reporting period. Publications that appeared during the last reporting period, or that have not yet been addressed in previous deliverables, relevant to this section are: [19, 22, 20, 10, 18, 36, 39] on spatio-temporal model checking for closure spaces, [40, 34, 2] on Spatial Signal Temporal Logic and [35, 37, 38, 21, 48, 13, 35] on scalable model checking techniques.

Spatio-temporal logics for closure spaces. *Topological spaces*, possibly enriched with metrics, or other spatial features (see [56]), are in many cases the mathematical structure of choice for the interpretation of spatial logics. The use of abstract structures has the advantage to separate logical operators, such as neighbourhood, from the specific nature of space (e.g., the number of dimensions, or the presence or absence of metric features, etc.). In [83], *closure spaces*, which generalise topological spaces, are proposed as a unifying approach treating both topological spaces and discrete structures such as graphs in a satisfactory way, including those identified as relevant to CAS in the deliverables of WP2. Finite spaces and graphs are subclasses of closure spaces — technically, they belong to the class of *quasi discrete closure spaces* — and the graph-theoretical notion of neighbourhood coincides with the notion of neighbourhood defined in the context of closure spaces.

Following up on the work by Galton and that of Smyth and Webster [115], in [68] Ciancia et al. proposed the logic Spatial Logic for Closure Spaces (SLCS), extending the topological semantics of modal logics to *closure spaces*. This framework provides a set of useful basic abstract spatial operators (closure, interior, boundary and others) that provide a structured way to define higher level spatial logical operators. In particular, SLCS is equipped with two *spatial operators*: a "one step" modality, called "near" and denoted by $\mathcal{N}$, turning the closure operator $\mathscr{C}$ into a logical operator, and a binary *spatial until* (or *surrounded*) operator $\mathcal{S}$, which is a spatial counterpart of the temporal *until*[1] operator. A point x satisfies formula $\mathcal{N}\phi$ if x is an element of the closure of the set of points satisfying ϕ. For formulas of the form $\phi_1 \mathcal{S} \phi_2$, the basic idea is that point x satisfies $\phi_1 \mathcal{S} \phi_2$ whenever there is "no way out" from ϕ_1 except passing by a point that satisfies ϕ_2. Moreover, the abstract spatial operators are suitable for the development of efficient spatial and (branching time) spatio-temporal model-checking algorithms in which these same closure space based operators play a role as well. Furthermore, metrics and distance functions can be added in an orthogonal way providing further spatial richness. We refer to Deliverable 2.1, Deliverable 3.1 and Internal Report 3.1 for a more detailed introduction to the theoretical background on the approach. Follow-up work on this line of research, performed in the last reporting period, is described in Section 2.3.

Spatial Signal Temporal Logic. *Signal Spatio-Temporal Logic* (SSTL) is an extension of Signal Temporal Logic [81, 103] with two spatial modalities. The first one, the *bounded somewhere* operator $\diamondsuit_{[w_1,w_2]}$ is taken from [106], while the second one, the *bounded surround* operator $\mathcal{S}_{[w_1,w_2]}$, is inspired by SLCS [68]. The logic comes with a boolean and quantitative semantics which can be found in [106, 40]. The boolean semantics defines when a formula is satisfied, the quantitative semantics provides an indication of the *robustness* with which a formula is satisfied [46, 34], i.e. how susceptible it is to changing its truth value for example as a result of a perturbation in the signals.

SSTL is interpreted on spatio-temporal, real-valued signals. Space is discrete and described by a weighted graph $G = (L, E, w)$, while the time domain $\mathbb{T}$ will usually be the real-valued interval $[0, T]$, for some $T > 0$. A spatio-temporal trace is a function $\mathbf{x} : \mathbb{T} \times L \to \mathbb{D}$, where $\mathbb{D} \subseteq \mathbb{R}^n$ is the codomain of the trace. As for temporal traces, we write $\mathbf{x}(t,\ell) = (x_1(t,\ell), \cdots, x_n(t,\ell)) \in \mathbb{D}$, where each $x_i : \mathbb{T} \times L \to \mathbb{D}_i$, for $i = 1, ..., n$, is the projection on the i^{th} coordinate/variable.

[1] Strictly speaking, the *unless* operator.

Spatio-temporal traces can be obtained by simulating a stochastic model or a deterministic model, i.e. specified by a set of differential equations. In [106] the framework of patch-based population models is discussed, which generalise population models and are a natural setting from which both stochastic and deterministic spatio-temporal traces of the considered type emerge. An alternative source of traces are measurements of real systems.

Spatio-temporal traces are converted into spatio-temporal boolean or quantitative signals. Similarly to the case of STL, each *atomic predicate* μ_j is of the form $\mu_j(x_1,\ldots,x_n) \equiv (f_j(x_1,\ldots,x_n) \geq 0)$, for $f_j : \mathbb{D} \to \mathbb{R}$. Each atomic proposition gives rise to a spatio-temporal signal. In the boolean case, one may define function $s_j : \mathbb{T} \times L \to \mathbb{B}$; given a trace $\mathbf{x}$, this gives rise to the boolean signal $s_j(t,\ell) = \mu_j(\mathbf{x}(t,\ell))$ by point-wise lifting. Similarly, a quantitative signal is obtained as the real-valued function $s_j : \mathbb{T} \times L \to \mathbb{R}$, with $s_j(t,\ell) = f_j(\mathbf{x}(t,\ell))$. When the space L is finite a spatio-temporal signal can be presented as a finite collection of temporal signals. Follow-up work on this research performed in the last reporting period is described in Section 2.4.

Scalable verification. Fluid flow approximation has shown to be a very successful approach to analyse properties of large population models in an efficient and scalable way. A brief introduction to this approach can be found in Deliverable 3.1. The idea to combine a fluid flow approach with model checking in order to verify properties of a single object in the context of a large population has first been described in [60], leading to Fluid Model Checking. This work has been completed with a detailed description of *global* fluid model checking algorithms and correctness proofs in the context of a CSL based logic and CTMC based population models and is now published in [35].

A different approach has been followed in the development of an on-the-fly approximated mean field model checking technique. Its foundations have been developed during the first year of the project (see the overview in Deliverable 3.1). In this approach we consider a model for interacting objects, where the evolution of each object is given by a finite state discrete time Markov chain (DTMC). A simple language for system specifications has been provided where the behaviour of a generic object of a system can be defined by means of two sets of defining equations, namely state definitions and action probability function definitions. These probabilities may depend on the global occupancy measure of the population model. Properties of a single object can be specified as bounded PCTL formulas. Also this work has been completed with efficient on-the-fly model checking algorithms and correctness proofs and has been published in [37]. The algorithm has been implemented in the prototype mean field model checker FlyFast, that was described in Deliverable 5.2. The implementation of FlyFast is based on the efficient on-the-fly probabilistic model checking approach described in [99]. FlyFast has been applied to various case studies, among which an extended bike sharing model [48], that was briefly discussed in Deliverable 3.2.

Section 2.5 describes further work on scalable verification performed during the last reporting period.

2.2 Objectives for the Reporting Period

There were several objectives for the work on scalable spatial and fluid/mean model checking. We briefly recall the objectives from the Description of Work and from Deliverables 3.1, 3.2 and the Internal Report 3.1 for this last reporting period concerning this topic.

- Development of scalable spatial and spatio-temporal model checking techniques that can be used in combination with mean field based analysis of large collective adaptive systems, but also to analyse snap-shot based spatial simulation traces (see Sect. 2.3).
- In spatial model checking it may be important to be able to specify spatial properties concerning *groups* of points in space rather than of individual points. For example, the property that agents associated to points in space are able to connect to one another and act as a group, or that they are located all together in a protected environment, or that they can share part of the same route to reach a common exit or goal. In all such situations, it is important to be able to predicate over spatial aspects, and possibly find methods to certify that a given collective adaptive system satisfies specific requirements in this respect (see Sect. 2.3).

$$
\begin{array}{rcll}
\Phi & ::= & \top & [\textsc{True}] \\
 & | & p & [\textsc{Atomic predicate}] \\
 & | & \neg\Phi & [\textsc{Not}] \\
 & | & \Phi \vee \Phi & [\textsc{Or}] \\
 & | & \mathscr{N}\,\Phi & [\textsc{Near}] \\
 & | & \Phi\,\mathscr{S}\,\Phi & [\textsc{Surrounded}] \\
 & | & \mathtt{A}\,\varphi & [\textsc{All Futures}] \\
 & | & \mathtt{E}\,\varphi & [\textsc{Some Future}] \\
 & & & \\
\varphi & ::= & \mathscr{X}\,\Phi & [\textsc{Next}] \\
 & | & \Phi\,\mathscr{U}\,\Phi & [\textsc{Until}]
\end{array}
$$

Figure 1: STLCS syntax

- Although the spatio-temporal model checking algorithms do work fine, there is much scope for further optimisations that are specific of spatial models, for example exploiting notions of spatial bisimulation and spatial aggregation to reduce the size of the space that needs to be verified (see Sect. 3.3).
- Further work on robustness analysis for SSTL including the development of prototype tools for spatio-temporal model checking and monitoring making the approach more widely available for application on CAS (see Sect. 2.4).
- Further work is foreseen in linking the model checking techniques to the modelling language CARMA developed in the context of WP4 and on extensions of the logic implemented in FlyFast (see Sect. 2.5).

2.3 Achievements Concerning Spatio-temporal Model Checking

Spatio-temporal model checking for closure spaces. We have investigated a combination of the temporal logic *Computation Tree Logic* (CTL) with SLCS, resulting in the *Spatio-Temporal Logic of Closure Spaces* (STLCS). In STLCS spatial and temporal fragments may be mutually nested. STLCS is interpreted on a variant of Kripke models, where valuations are interpreted at points of a closure space. Fix a set *AP* of proposition letters. STLCS formulas are defined by the grammar shown in Fig. 1, where p ranges over *AP*. The logic features the CTL path quantifiers `A` ("for all paths"), and `E` ("there exists a path"). As in CTL, such quantifiers must necessarily be followed by one of the path-specific temporal operators, such as[2] $\mathscr{X}\Phi$ ("next"), $\mathtt{F}\Phi$ ("eventually"), $\mathtt{G}\Phi$ ("globally"), $\Phi_1\,\mathscr{U}\,\Phi_2$ ("until"), but unlike CTL, in this case Φ, Φ_1 and Φ_2 are STLCS formulas that may make use of spatial operators 'near' ($\mathscr{N}$), 'surrounded' ($\mathscr{S}$) and operators derived thereof (see Internal Report 3.1 for further details and examples).

A model checking algorithm for STLCS has been defined in [18], which is a variant of the classical CTL labelling algorithm [69, 44], augmented with the algorithm in [68] for the spatial fragment. The algorithm, which operates on finite spaces, has been implemented as a prototype [85], which is described in [18]. The same algorithm is also implemented in the tool `topochecker` [66].

The tool has been applied (in its various implemented versions) on two case studies of the project. In [17], which extends the work in [67], further properties of vehicular movement in public transport systems have been analysed. In particular a phenomenon known as *clumping* has been addressed. Clumping (also known as platooning) may occur in so-called "frequent" services – those where a timetable is not published. Clumping occurs where one bus catches up with – or at least comes too close to – the bus which is in front of it. In the absence of a published timetable for frequent services the important performance metric to consider is not timetable adherence but headway, a measure of the separation between subsequent buses. This separation can be defined both in terms of distance between buses on the same route or in terms of the time between two buses

[2] Some operators may be derived from others; for this reason in Fig. 1 we use a minimal set of connectives. As usual in logics, there are several different choices for such a set.

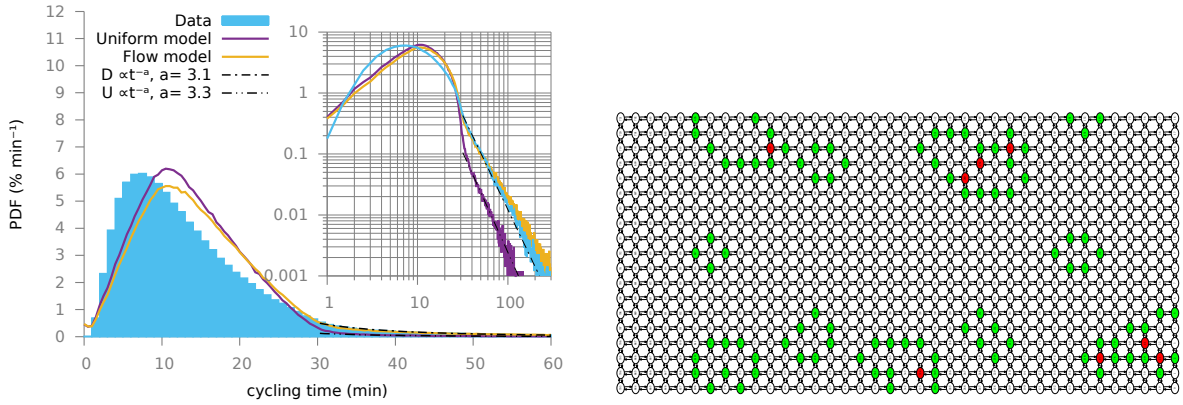


Figure 2: (left) Cycling duration histograms (Data) in London, using 831,754 trip records in October 2012, and results of simulation of the uniform model (magenta lines) and the flow model (orange lines). Maintenance trips are not considered. (right) Stations of state 80 of the simulation that are on the boundary of the region of points that will eventually become a full station cluster (green), and stations that, whenever they are full, stay full until the end of the simulation trace and become part of a cluster (red).

on the same route passing by the same bus stop. These two different notions of clumping can be characterised formally using STLCS on a time series of street map images on which the bus positions are projected. In [36] STLCS was used to check spatio-temporal properties of bike sharing systems. In particular, in [36] spatio-temporal model checking has been used to detect the emergent formation of 'clusters' of full (and empty) stations in the simulation traces of a Markov Renewal Process (MRP) model of large bike sharing systems [39], e.g. as large as the one in London. The MRP model describes the dynamics of a population of agents, modelling bike sharing users, coupled with the dynamics of bicycle stations located in a two-dimensional rectangle representing a city. Agent behaviour has been modelled based on basic human factors such as mean walking and biking speed, but also integrating simple decision models that define the area in which a users looks for a suitable parking place close to its selected destination. The model covers all parts of a bike sharing trip, namely both the parts on foot and those by bike.

Interestingly, the simulation traces of the MRP model show a good correspondence with observed cycling times in the real system (see Fig. 2). As in the real data, the cycling time distribution observed in the model shows an unexpected feature, namely the tail of the distribution is algebraic. This indicates that there are relatively many cases in which bike users return their bikes after much longer time than could be expected given the 30 minutes free allowance, and do so in a sort of 'accidental' or 'unforeseen' way. The hypothesis is that these late returns could be explained by the problems that users experience in returning their bikes when no parking places are available close to their desired destination. A model in which users are moving randomly from one place to another, using the bike sharing system, but always with the aim to return their bike within 30 minutes to a station, shows much fewer users with a late return. However, to reproduce the heavier algebraic tail it turned out to be sufficient to increase congestion in the bike sharing system by introducing flows of commuters that need to be in a populated place at a certain time. Using spatio-temporal model checking it was shown that the introduction of the flows also leads to the emergence of *clusters of full stations* which are a plausible explanation for later-than-expected returns. Clusters were not detected in traces obtained from the random model without commuter flows (i.e. with only agents moving randomly from some origin to some destination).

A tutorial introduction to the theory of closure spaces, spatial and spatio-temporal logics and additional examples was presented at the 16th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Quantitative Evaluation of Collective Adaptive Systems school in Bertinoro and can be found in [20].

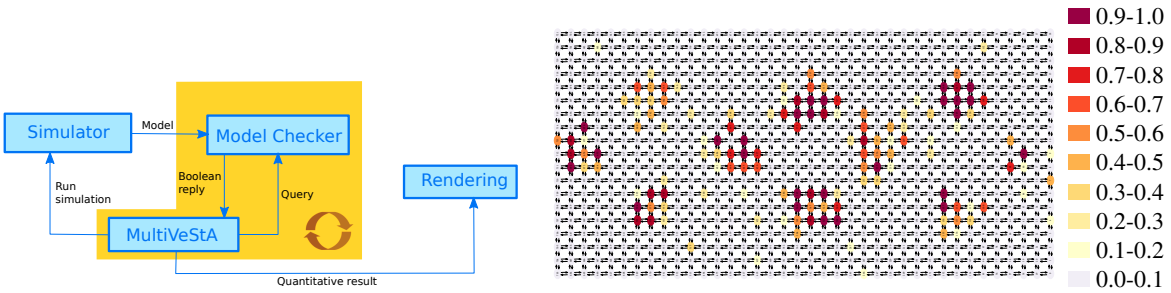


Figure 3: (left) Collaboration in the tool-chain used for statistical spatio-temporal model checking. (right) Probability that a user intending to park her bike in a station finds it full and cannot find a parking place within three consecutive attempts in neighbouring stations for a artificial pattern of bikes request and return (see [22] for further details).

Combining statistical and spatio-temporal model checking. The stochastic simulator for the MRP BSS model, briefly discussed above, has in turn been combined with both the spatio-temporal model checker `topochecker` and the tool MultiVeStA [114][3]. The latter is a tool for distributed statistical model checking and can be easily integrated with any existing discrete event simulator, or formalism that provides probabilistic simulation. In [22] we have used MultiVeStA to estimate quantitative spatio-temporal properties of bike sharing systems. The methodology aims to estimate the likelihood, at each point of space, that a given formula (with boolean valuation) is true, with a user-specified *global confidence interval* – that is, the same interval is used for all points. In this way, a *heat-map* is produced that associates to each point of space a probability value.

The approach exploits the "multi" in MultiVeStA, by reusing the same simulation trace *for each point of the space*, resulting in a large number of random variables – one for each point of space and formula – being analysed at once. The resulting collaboration pattern is depicted in Figure 3. We remark that the total execution time for all the properties we consider is in the order of around five hours on a standard laptop; this hints at the importance of observing multiple points at the same time (exploiting the specific capabilities of MultiVeStA); the size of the considered space is 722 points and involving 2400 agents (of which 2000 modelling commuters) reflecting a BSS of the size of that of London. Running the statistical model checking sequentially for each point in space would multiply the execution times accordingly, changing the approach from "feasible" to "infeasible".

Additional spatial operators. In [19] the spatial logic SLCS has been consolidated and extended with a further operator, $\mathscr{P}$, capturing the notion of spatial propagation; intuitively the formula $\phi\,\mathscr{P}\,\psi$ describes a situation in which the points satisfying ψ can be reached by paths rooted in points satisfying ϕ and, for the rest, composed only of points satisfying ψ. Furthermore the logic has been extended with operators for *collective properties*, namely properties which are satisfied by *connected sets* of points, rather than points in isolation. The formal semantics of the extended logic—CSLCS, Collective SLCS–are provided in the form of a satisfiability relation defined using the notion of infinite path in closure spaces. The model-checking algorithms for spatial model checking have been extended in order to treat the newly introduced operators and related correctness proofs have been provided. The algorithms have been applied to various examples from the domain of collective adaptive systems, among which emergency egress [19], using a prototype implementation of the spatial model-checker.

The satisfaction relation of the logic for each collective formula ψ is given in the form $\mathscr{M},A \models_C \psi$, where $\mathscr{M}$ is a closure model, and $A \subseteq X$ is a set of points.

Definition 2.1. *Given a model $\mathscr{M} = ((X,\mathscr{C}),\mathscr{V}))$, and $A \subseteq X$, collective satisfaction $\models_C$ is given by the inductive definition below, where $\models$ is the individual satisfaction relation:*

[3] Available at `http://sysma.imtlucca.it/tools/multivesta/`

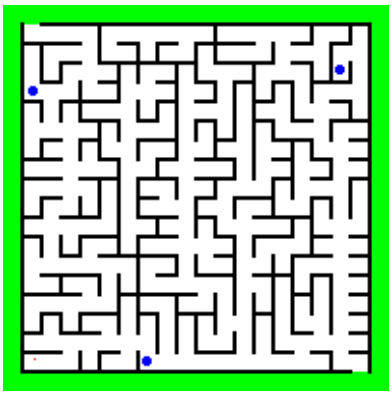


Figure 4: Blue circles are *not* able to reach the *same* exit.

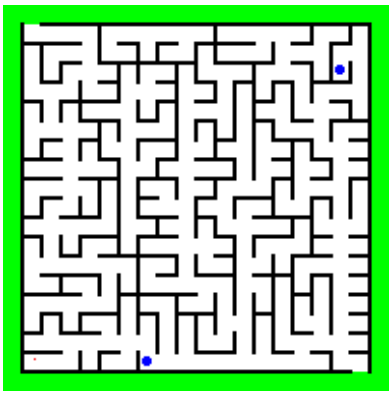


Figure 5: Blue circles are able to reach the *same* exit.

$$
\begin{array}{lll}
\mathscr{M},A \models_C \top & & \\
\mathscr{M},A \models_C \neg\psi & \iff & \mathscr{M},A \not\models_C \psi \\
\mathscr{M},A \models_C \psi_1 \wedge \psi_2 & \iff & \mathscr{M},A \models_C \psi_1 \textit{ and } \mathscr{M},A \models_C \psi_2 \\
\mathscr{M},A \models_C \phi \prec \psi & \iff & \mathscr{M},\{x \in A \mid \mathscr{M},x \models \phi\} \models_C \psi \\
\mathscr{M},A \models_C \mathscr{G}\phi & \iff & \exists B \subseteq X.A \subseteq B \wedge B \textit{ is path-connected} \wedge \\
 & & \forall z \in B.\mathscr{M},z \models \phi
\end{array}
$$

Let ϕ be an individual formula, and ψ a collective formula. Informally, $\phi \prec \psi$ (read: ϕ *share* ψ) is satisfied by set A when the subset of points of A satisfying the individual property ϕ also satisfies the collective property ψ. Formula $\mathscr{G}\phi$ holds on set A when its elements belong to a *group*, that is, a possibly larger, path-connected set of points, all satisfying the individual formula ϕ. The definition of $\mathscr{G}$ requires the existence of a set B which is possibly larger than A. The intuition is that the elements of A are part of a larger "collective", consisting of elements satisfying ϕ.

We considered variants of connectedness as the most basic forms of *collective* and *spatial* property. In particular, we used path-connectedness, in line with the path-based interpretation of SLCS provided in [19]. Connectedness is "collective" in the sense that it is not merely determined by a property of the singletons composing a set, and it is not even preserved in subsets of a connected set. On the other hand, even though one could imagine all sorts of collective predicates on a model, we focussed on (path-)connectedness, as it is completely determined by the structure of a closure space. For this reason, we considered it a fundamental collective property, deserving special treatment in the field of spatial logics, akin to the notion of transition in models of modal logics. Due to the restrictions that we introduced (mainly the strict layering of the collective and individual fragments) the logic CSLCS can be automatically verified at a computational cost which is comparable to that of SLCS. Using CSLCS one is able to check that given individuals lie in the *same* area of space, and they share specific properties. Informally (and depending on the chosen closure model), this idea can be interpreted, for example, as: the fact that certain individuals are able to connect and act as a group; that they may follow the same route to reach a goal; that they are located all together in a protected environment; etc.

Consider as an example the maze in Figure 4. The three blue circles in the maze can all reach an exit; however, they cannot "collectively" do so, as they cannot join and get out through the *same* exit. In Figure 5, on the other hand, the two blue circles *can* get out through the *same* exit. The formula $blue \prec (\mathscr{G}((blue \vee white)\,\mathscr{T}\,green))$ returns *false* in the first model, and *true* in the second one. The given formula, which is interpreted globally, asserts that all the blue points are part of a strongly connected component of points that can reach the (green) exit passing by points that are either blue or white.

Spatial operators for Medical Imaging. An unexpected innovation developed in the project is the application of spatial model checking in a completely different domain, namely that of Medical Imaging (MI). In [10] two kinds of additional operators have been proposed and experimented with: distance operators and texture operators. In MI many images are produced with magnetic resonance techniques producing 3D discrete images composed of so-called voxels, i.e. 3D volume pixels. The reference distance between two voxels is the Euclidean distance. Two different distance transform operators have been added to the set of spatial operators of STLCS: an error-free linear algorithm for Euclidean distance in regular grids based on Maurer et al. [104] and a distance transform based on a modified Dijkstra's shortest path algorithm [84] that can be used on general weighted graphs. Informally, given the set of points in an image that satisfy a property ϕ, the distance transform globally computes the distance, for any point in the space, to the nearest point x that satisfies ϕ. Texture analysis (TA) operators are designed for finding and analysing patterns in medical images, including some that are imperceptible to the human visual system. Patterns in images are entities characterised by brightness, colour, shape, size, etc. In TA, image textures are usually characterised by estimating some descriptors in terms of quantitative features. Typically, such features fall into three general categories: syntactic, statistical, and spectral. Combining texture and distance operators with SLCS a very interesting flexible, concise, unambiguous and modular logical language emerges for the composition of medical image analysis strategies providing analysts with much more freedom to formulate new strategies and exchange and discuss them with their colleagues.

First experiments with the extended spatial model checker shows promising results for example in identifying the different textures of a tumor and close oedema as shown in Fig. 6. Such distinctions are of great importance in the preparation of, for example, radiotherapy or surgery.

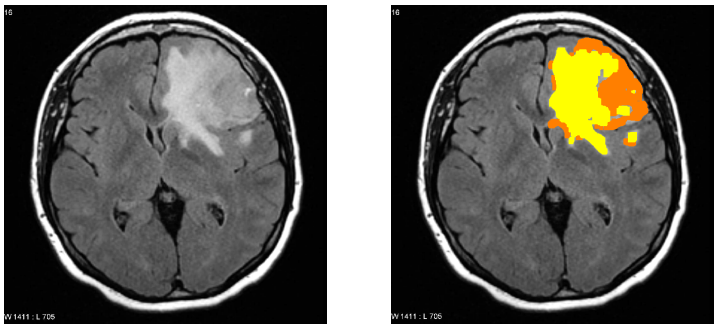


Figure 6: (left) A slice of a FLAIR MR acquisition of a brain affected by a glioblastoma. (right) Final result of application of threshold, distance, and the texture operator. The yellow area is the identified oedema; the orange area is the tumor (case courtesy of A.Prof Frank Gaillard, Radiopaedia.org, rID: 5292).

2.4 Achievements Concerning Spatio-temporal Monitoring

Signal Spatio-Temporal Logic (SSTL) is an extension of Signal Temporal Logic [81, 103] with two spatial modalities. We briefly recall the syntax of the Spatial Signal Temporal Logic (SSTL) which is given by

$$\varphi := \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathscr{U}_{[t_1,t_2]}\, \varphi_2 \mid \ \diamondsuit_{[w_1,w_2]}\varphi \mid \varphi_1 \,\mathscr{S}_{[w_1,w_2]}\, \varphi_2.$$

Atomic predicates, boolean operators, and the until operator $\mathscr{U}_{[t_1,t_2]}$ are those of STL. The spatial operators are the *somewhere* operator, $\diamondsuit_{[w_1,w_2]}$, and the *bounded surround* operator $\mathscr{S}_{[w_1,w_2]}$, where $[w_1,w_2]$ is a closed real interval with $w_1 < w_2$.

As briefly anticipated in Deliverable 7.2 and further developed in [40] and in Internal Report 3.1, SSTL can be used to identify the formation of *patterns* in a reaction-diffusion system, an example of which is shown in Fig. 7. In particular, in [40] the logic has been used to identify more or less circular 'spots' of a specified dimension. To identify the insurgence time of the pattern and whether it remains stable over time the spatial property has been combined with temporal operators of the logic.

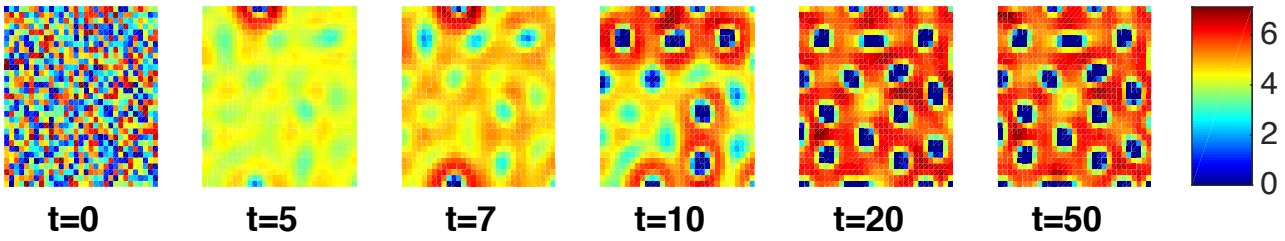


Figure 7: Value of x^A for the Turing system in [40] for $t = 0,5,7,12,20,50$ time units with parameters $K = 32, R_1 = 1, R_2 = -12, R_3 = -1, R_4 = 16, D_1 = 5.6$ and $D_2 = 25.5$. The initial condition has been set randomly. The colour map for the concentration is specified in the legend on the right.

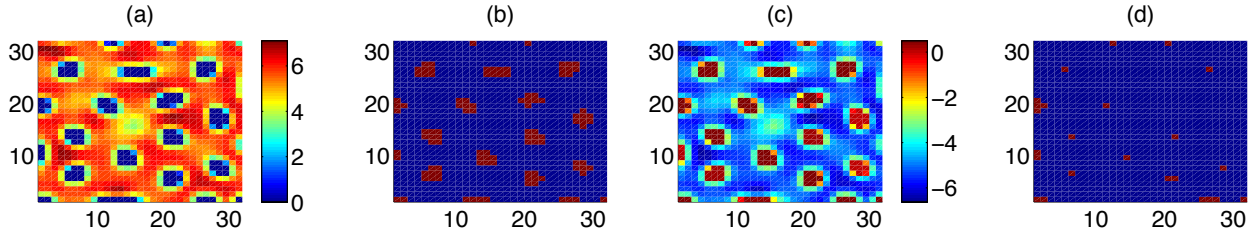


Figure 8: Validity of formula (2) with parameters $h = 0.5, T_{\text{pattern}} = 19, \delta = 1, T_{\text{end}} = 30, w_1 = 1, w_2 = 6$ for (b), (c) and $w_2 = 4$ for (d). (a) Concentration of A at time t = 50; (b) (d) Boolean semantics of the property ϕ_{pattern}; the cells (locations) that satisfy the formula are in red, the others are in blue; (c) Quantitative semantics of the property ϕ_{pattern}; The value of the robustness is given by a colour map as specified in the legend on the right of figure (c).

Spots with a low concentration of species A can be identified by points in space satisfying the following SSTL formula where x^A denotes the concentration of species A:

$$\phi_{\text{spot}} := (x^A \leq h) \mathscr{S}_{[w_1,w_2]} (x^A > h). \tag{1}$$

The use of distance bounds w_1 and w_2 in the surround operator allows one to constrain the size/diameter of the spot to $[w_1, w_2]$. To identify the insurgence time of the pattern and whether it remains stable over time the spatial property needs to be combined with temporal operators in the following way:

$$\phi_{\text{pattern}} := \mathscr{F}_{[T_{\text{pattern}}, T_{\text{pattern}}+\delta]} \mathscr{G}_{[0,T_{\text{end}}]} (\phi_{\text{spot}}); \tag{2}$$

ϕ_{pattern} means that eventually ($\mathscr{F}$) at a time between T_{pattern} and $T_{\text{pattern}} + \delta$ the property spot becomes true and remains true ($\mathscr{G}$) for at least T_{end} time units. Fig. 8(b) shows the validity of the property ϕ_{pattern} in each cell $(i, j) \in L$, for both the boolean and the quantitative semantics.

The qualitative and quantitative monitoring algorithms for SSTL has been implemented in a prototype tool developed in Java. It consists of a Java library (jSSTL API) and a front-end, integrated in ECLIPSE. Both the library and the ECLIPSE plugin are available from `http://quanticol.sourceforge.net/`. The source code is available from `http://bitbucket.org/LauraNenzi/jsstl`. The library can be used to integrate jSSTL within other applications and tools, whereas the ECLIPSE plugin provides a user friendly interface to the tool. Furthermore, the modular approach of the implementation allows one to develop different front-ends for jSSTL. A more detailed description of jSSTL can be found in Deliverable 5.3.

In [2], the authors combined SSTL with recent machine learning approaches to verification [13] and synthesis [46, 34], to analyse the robustness of stochastic spatio-temporal models and to design specific behaviours. The idea is to extend the semantics of SSTL to stochastic systems along the lines of [46, 34], by considering either the probability with which a formula is satisfied by a stochastic model, for the qualitative semantics, or the distribution of the robustness score, for the quantitative one. This allows one to combine SSTL with

statistical model checking tools. In particular, in [2] the authors consider a model of developmental biology, the french flag pattern formation, in which the spatial gradient of the Bicoid protein is responsible for the horizontal segmentation of the *Drosophila* embryo. Two different kinds of analysis are discussed in [2]

1. The robustness of the property encoding the french flag property (i.e. a decaying concentration of the signalling protein along the horizontal axis, identifying three regions, of high, medium and low expression), is studied with respect to the model parameters. To this end, the authors exploit a novel method, smoothed model checking [13], which allows for the statistical reconstruction of the satisfaction probability as a function of model parameters;

2. The model is optimally designed in order to maximise the robustness of the SSTL behavioral specification. The authors here use the statistical system design approach presented in [46, 34], which leverages a recent provably convergent Bayesian optimisation algorithm.

Moreover, the Java implementation of SSTL has been integrated in the tool U-check [62], which implements these novel statistical verification techniques. Examples of the application of jSSTL on the case studies of the project are provided in Deliverable 4.3.

A further extension of Signal Spatio-Temporal Logic is presented in [29]. This new spatio-temporal logic, called Three-Valued Spatio-Temporal Logic (TSTL), is a logic provided with a three-valued semantics that widens the analysis of properties of stochastic spatio-temporal systems. Starting from the estimation of satisfaction probabilities of given logical formulas, this extension allows us to verify properties of the spatio-temporal evolution of these estimated values. The additional layer of analysis embeds the uncertainty intrinsically related with these estimations, usually evaluated through simulation-based statistical methods.

2.5 Achievements Concerning Scalable Verification

Fluid Model Checking. The work on Fluid Model Checking has been extended to consider checking of properties, of individual agents, which incorporate rewards [12]. The algorithms and the convergence results have been presented, and the approach has been applied to a bike sharing example. In particular, individual agent models have been extended with reward structures which may be comprised of a state reward function and a transition reward function. The first function gives the non-negative reward of spending one time unit in any state, while the second encodes the non-negative reward for taking a certain transition. Four kinds of reward expressions have been considered: cumulative rewards up to time T; instantaneous rewards at time T; steady state rewards and bounded reachability rewards. These reward properties have been added to the time-bounded fragment of CSL. For each type of reward property a global model checking algorithm has been provided and a convergence theorem has been proved. The latter shows that in the limit of the population size N, the obtained reward in the full model is equal, with probability 1, to that obtained with the fluid approximation of the model.

An example of the use of rewards in bike sharing is to model user dissatisfaction. Dissatisfaction with the service, is incurred whenever the member is not able to obtain a bike ('fail acquire') or return ('fail return') a bike on the first attempt. This is captured by the reward structure with a transition reward equal to d for the actions 'fail acquire' and 'fail return'. The reward is set 10 times higher if the user fails to acquire or return the bike a second time. The instantaneous value of the reward will give the expected level of dissatisfaction of the member, which can be required to be below a certain threshold r. The performance of the algorithms have been compared with that of statistical model checking, and the fluid model checking approach has been shown to take times three orders of magnitude smaller for the examples considered. Moreover the checking time, in the case of fluid approximation, is independent of the size of the population, although the performance is sensitive to the time bounds considered in the property. The quality of the approximation has been shown to be good, but also dependent on the kind of property considered. For further details we refer to [12].

Mean Field Model Checking. In [38] it is shown that, under suitable convergence and scaling conditions, fluid model checking of bounded CSL formulas on selected individuals in a continuous large population model can be approximated by checking equivalent bounded PCTL formulas on corresponding objects in a discrete

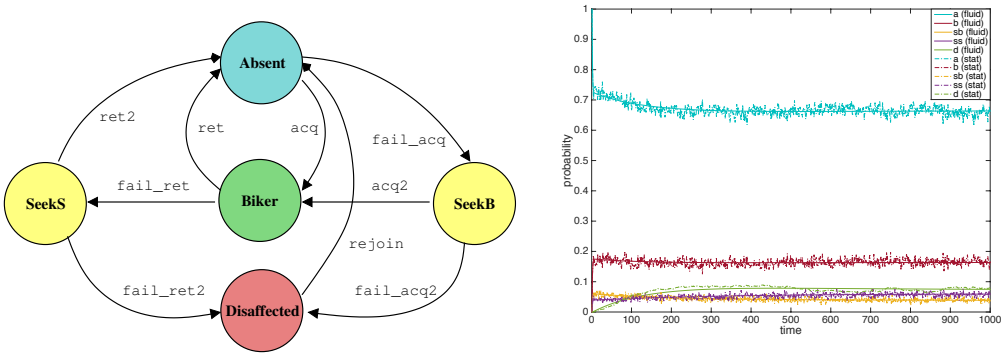


Figure 9: States and transitions of a single member in the bike sharing system (left). Comparison of the fluid approximation solution $z(t)$ with the statistical estimation (5000 runs) of the state probabilities for $Z^{(N)}(t)$. Parameters are $N = 300$, $S = 150$, $B = 100$, $k_{\mathtt{acq}} = 0.25$, $k_{\mathtt{acq2}} = 2$, $k_{\mathtt{ret}} = 1$, $k_{\mathtt{ret2}} = 2$, $k_{\mathtt{rej}} = 0.005$, $h = 0.05$, $q = 0.1$, $X_a(0) = N$, $X_i(0) = 0$, for $i \neq a$.

time, clock-synchronous Markov population model, using an on-the-fly mean field approach. The proposed technique is applied to a benchmark epidemic model and a client-server case study showing promising results also for the challenging case of nested formulas with time dependent truth values. The on-the-fly results obtained with the prototype mean field model checker FlyFast are compared to those obtained via global fluid model checking [35] and statistical model-checking approaches.

During the third reporting period FlyFast has been further improved and applied on various case studies, including the benchmark applications regarding a computer worm epidemic, a client-server system [38] and a push-pull gossip protocol [26]. The latter is described in more detail in Deliverable 5.3 illustrating the use of the FlyFast model checker. In Deliverable 4.3 FlyFast has been used in combination with CaSL to illustrate the modelling and mean field model checking of properties of a bike sharing system.

Furthermore, the FlyFast front-end modelling language has been extended in order to deal with *components* and *predicate-based interaction* [21]. This extension has been inspired by CARMA [59, 102] (see Deliverable 4.2 and Deliverable 4.3). Components are expressed as *process-store* pairs; actions are *predicate based multi-cast* output and input primitives. Associated to each action there is also an (atomic) probabilistic store-update. For instance, assume components have an attribute named loc which takes values in the set of points of a space type. The following action models a multi-cast[4] via channel α to all components in the same location as the sender, making it change location randomly: $\alpha^*[\mathsf{loc} = \mathbf{my}.\mathsf{loc}]\langle\rangle\{\mathsf{loc} \leftarrow \mathsf{randomLoc}(\mathsf{loc})\}$. Here randomLoc is assumed to be a random generator of points in the space. The computational model is *clock-synchronous* (as in FlyFast,) but at the component level. In addition, each component is equipped with a local *outbox*. The effect of an output action $\alpha^*[\pi_r]\langle\rangle\sigma$ is to deliver output label $\alpha\langle\rangle$ to the local outbox, together with the predicate π_r, which receiver components will be required to satisfy, as well as the current store γ of the component executing the action; the current store is updated according to update σ. Note that output actions are *non-blocking* and that successive output actions of the same component rewrite its outbox. An input action $\alpha^*[\pi_s]()\sigma$ by a component will be executed with a probability which is proportional to the *fraction* of all those components whose outboxes currently contain the label $\alpha\langle\rangle$, a predicate π_r which is satisfied by the component, and a store γ which satisfies in turn predicate π_s. If such a fraction is zero, then the input action will not take place (input is blocking), otherwise the action takes place, the store of the component is updated via σ, and its outbox cleared. Thus, as in the original FlyFast language, component interaction is probabilistic, but now the *fraction* of the components *satisfying the relevant predicates* plays a role in the computation of transition probabilities. The formal probabilistic semantics of the extended language has been provided and a translation to the original FlyFast language which makes the model-checker support the extended language. The translation has been proven correct.

[4] *Multi-cast* interaction is denoted using the _* notation, as in CARMA.

Model Checking Linear Noise Approximations. In [11] we considered how to exploit stochastic approximation to model check time-bounded global reachability properties of a Markov population model, described in the biochemical reaction style. Global reachability properties are at the heart of verification algorithms for many formal specification languages, including the temporal logics CSL and LTL and deterministic automata. The idea in this paper is to approximate the population model by a *continuous state space* stochastic process by means of the *Linear Noise Approximation (LNA)*. In this way we obtain a Gaussian process which approximates the population model [123]. This approximation is efficient when we are interested in the single time marginal distributions, as they are Gaussian and can be obtained by solving a set of ODEs for the mean and the covariances. This was exploited in [58] to obtain an efficient verification procedure for special classes of local to global properties which look at a fraction of individuals satisfying a local specification. These ideas, however, do not work for global reachability properties, which are genuine path properties, and require knowledge about the probability distribution over paths. In theory, LNA defines a path distribution over càdlàg functions from time to the continuous state space, but computing this distribution is a challenging task, as the process is in continuous space.

The breakthrough of [11] is to observe that we can leverage special properties of Gaussian Processes to construct an efficient discretisation in space and time of the continuous process, which can be solved efficiently by standard algorithms for Discrete Time Markov Chains (DTMC). We consider polytope regions specified by a set of k linear inequalities, with k small (typically equal to 1 or 2 in practical applications), and use these inequalities to construct a linear projection. Applying the projection to the Gaussian Process (GP) obtained by LNA, we still get a smaller Gaussian process, typically one or two dimensional, yet depending on the time-dependent average and covariance of the initial process. The next step consists in identifying a suitable discretisation of time, by fixing a step-size h, and to compute the time-dependent transition kernel of the discrete-time Markov process obtained from the original GP. This can be done efficiently as the solution of a set of suitably derived differential equations, see [11] for mathematical details. Finally, space is also discretised, computing the transition probability matrix of the final DTMC by averaging the kernel for each pair of discrete states. Once the discretisation is constructed, it can be used to compute global reachability properties by standard DTMC algorithms. In [11], we report results on a few examples taken from computational systems biology, showing good accuracy and a considerable computational gain.

As a small illustration of the method, assuming that we have a population called $Lp3$, one can express the global property that "the average probability over the first 10 seconds that the population size of L3p exceeds 40 is 0.3" as follows:

$$P_{>0.3}[Lp3, [40, \infty]]_{[0,10]}$$

We also prove the asymptotic correctness of the method, combining linear noise convergence with convergence results for the discretisation steps.

Statistical Model Checking of Uncertain Systems. In [13], we focus our attention on statistical methods for model checking, which are generally much more scalable than standard numerical methods. Statistical Model Checking (SMC [97]) is a randomized algorithm which leverages the possibility of drawing a large number of samples from generative models such as Markov Chains. By repeatedly drawing independent samples (runs/ trajectories) from the model the satisfaction probabilities of linear time properties are estimated as averages of satisfactions on individual runs; by the law of large numbers, these averages will converge to the true probability in the limit when the sample size becomes large, and general asymptotic results permit us to bound (probabilistically) the estimation error.

Both analytical and statistical tools for model checking however start from the premise that the underlying mathematical model is fully specified (or at least that a mechanism to draw independent and identically distributed samples exists in the case of SMC). This is both conceptually and practically problematic: models are abstractions of reality informed by domain expertise. Condensing the domain expertise in a single vector of parameter values is at best an approximation, while parametric uncertainty is the norm.

Therefore, is it very relevant to have methods to reason about how the qualitative properties of a system vary while varying model parameters. In [13], we develop an efficient approach to this end, leveraging Bayesian ma-

chine learning ideas to build estimates of the functional dependency of the satisfaction probability with respect to model parameters, providing also error bounds on the estimate. The first step is the definition the satisfaction function of a Metric Interval Temporal Logic formula, the natural extension of the concept of satisfaction probability of a formula to the case of Markov Population Models with parametric uncertainty. We prove that, under mild conditions, such a satisfaction function is a smooth function of the uncertain parameters of the population model. We then show how the satisfaction function can be approximated arbitrarily well by a sample from a Gaussian Process (GP) [112], a non-parametric distribution over spaces of functions, and use the GP approach to obtain an analytical approximation to the satisfaction function. This enables us to predict the value (and related uncertainty) of the satisfaction probability at all values of the uncertain parameters from individual model simulations at a finite (and generally rather small) number of distinct parameter values. This approach has been called "smoothed model checking". In [13], the method has been tested on three non-trivial examples, showing how smoothed model checking can provide an accurate estimation of the satisfaction function. Moreover, even if we are interested only in a predefined set of points of the parameter space, smoothed model checking is still superior to standard statistical model checking, obtaining a comparable accuracy with about one order of magnitude less simulations overall. This is due to the fact that GP learning algorithms, suitably tailored to the binomial noise model underlying the repetition of simulations and checking of property satisfaction in a point of the parameter space, are capable of transferring information from neighbouring parameter values, thus reducing the need for deep sampling at each value of the set of parameters of interest. This method is, up to now, the most efficient approach for parametrised verification [93], capable of easily scaling to large population models and to explore efficiently up to 5-6 parameters a once.

3 Model Reduction Techniques

As with most large-scale systems, the evaluation of quantitative properties of collective adaptive systems is an important issue that is crosscutting all its development stages, from design (in the case of engineered systems) to runtime monitoring and control. Unfortunately it is a difficult problem to tackle in general, due to the typically high computational cost involved in the analysis. This calls for the development of appropriate quantitative abstraction techniques that preserve most of the system's dynamical behaviour using a more compact representation. This section reports on the achievements concerning abstraction and reduction for both CTMCs and ODEs.

3.1 Background

Previous work on the topic of model reduction was discussed in Deliverable D3.2. In particular, a method was established to identify and remove populations that have no significant impact on a measure of interest during simulation-based analysis [82] and a method which automatically derives ODEs to capture the moments of population CTMCs and carries out moment-closure analysis to reduce the number of ODEs which must be considered [23]. Furthermore, a number of simplification approaches based on the notions of behavioural equivalence were presented for the fluid semantics of the process algebra PEPA [120, 92] and for chemical reaction networks [64]. However, the latter approaches suffered from the limitations of being language-specific and not *complete*, in the sense that the criteria for equivalence turned out to be only sufficient conditions for aggregation in general.

3.2 Objectives for the Reporting Period

In this deliverable we report on work that has addressed some of the aforementioned limitations, in line with the plan of future work outlined in D3.2:

1. The development of abstraction techniques for stochastic processes (cf. Section 3.3).
2. The full characterisation of equivalence relations for ODEs, thus identifying necessary and sufficient conditions for aggregation (cf. Section 3.4).

3. The extension to a domain-agnostic notion of abstraction/aggregation (cf. Section 3.4).

3.3 Achievements Concerning Abstractions for Stochastic Processes

There is a long tradition of equivalence relations being used to partition and quotient the state space of state-based models to obtain a more compact representation which captures equivalent information. For example for stochastic process algebras, Markovian bisimulation and strong equivalence have been shown to characterise ordinary lumpability in the underlying CTMC, leading to a reduced model which can be solved more efficiently [91, 90, 57].

In [31], the authors explored the extent to which these ideas can be carried over to the richer modelling languages developed within the QUANTICOL project. PALOMA was chosen as the focus for the study as it incorporates several of the novel features identified as important in modelling collective adaptive systems — attribute-based communication, location, broadcast and unicast capabilities — without the full generality of CARMA. As shown in [31], the notion of Markovian bisimulation as generally considered, if applied naively, is too strong, leaving little opportunity for a notion of equivalence that is not isomorphism. Instead the authors considered equivalence of a component within the context of a given system. This supports the idea of being able to substitute one component, perhaps with a more efficient implementation, for another within a given system even though they may not exhibit exactly the same behaviour in arbitrary contexts. The authors also sought some flexibility in the consideration of the spatial aspects of behaviour leading to a notion of equivalence that captures the relative positions of components, rather than their absolute locations. Full details of the work are presented in [109].

In [30] the authors consider a novel approach to model aggregation for CTMCs, the mathematical model that underlies stochastic process algebraic languages such as PEPA, PALOMA and CARMA. Aggregation seeks to identify sets of states that may be lumped into a single macro state, allowing a more compact model to be analysed. Strict conditions such as lumpability are required if the Markov property is to be preserved in the CTMC, but in some cases reasonable results can be obtained without the stringent conditions that lumpability imposes with respect to state probabilities and transition rates. In the approach of [30], rather than consider the quantitative characteristics of states at the CTMC level, the authors seek to identify macro-states through consideration of high-level properties in terms of the behaviour of the system. Thus the satisfiability of a set of temporal logic formulae is used to identify sets of states that share common behavioural properties and macro-states are based on these sets.

Whilst theoretically feasible, such an approach, if applied directly, would be computationally infeasible since model checking of the formulae would need to be carried out at every state in the state space before the aggregation could be formed. Instead the authors propose an approach which relies on sampling only a subset of states and then applying the smoothed model checking result of Bortolussi *et al.* [13], to extrapolate values in the remaining states using Gaussian Process emulation. To support the clustering into aggregates the resulting data is subjected to multi-dimensional scaling, a dimensionality reduction technique that optimally preserves distances in non-Euclidean spaces. Through this approach macro-states can be defined which behave coherently with respect to the logical specifications.

In [28] Bisimulation of Labelled State-to-Function Transition Systems (Labelled FuTS) is revisited from a coalgebraic perspective. A correspondence result is established stating that FuTS-bisimilarity coincides with behavioural equivalence of the associated functor. The FuTS framework [78] has been used as a unifying approach to the definition of the operational semantics of major process languages, ranging from stochastic process languages like PEPA, to a language for Interactive Markov Chains, a (discrete) timed process language and a language for Markov Automata. The equivalences underlying these languages have been related to the bisimilarity of their specific FuTS. By the correspondence result a coalgebraic justification of the equivalences of these calculi is obtained. The operational semantics of the CARMA language presented in Deliverable 4.2 has also been given using the FuTS framework.

Finally, in [27], a state-space reduction procedure for optimising the translation from the attribute-based process language proposed in [21] to FlyFast has been presented, which is based on probabilistic bisimulation where transition probabilities are functions of occupancy measure vectors. The specific syntax of the probability

function definitions of the attribute based process language guarantees decidability of equality of cumulative probabilities so that (suitable customisation of) standard state reduction algorithms can be used.

3.4 Achievements Concerning Abstractions for Ordinary Differential Equations

Even when the original description of a population model is a CTMC, the underlying behaviour is characterised by a (large-scale) system of ODEs (i.e., the forward equations of motion of the probability distribution). This section overviews recently developed techniques (reviewed in [32] and [33]) that consider the ODE abstraction problem from an algorithmic viewpoint [16, 14], with the intent of computing reduced systems with the following properties:

P1. The abstraction should come with formal guarantees on the relationship between the abstract dynamics and the original one. This enables the modeller to use the abstract model with full confidence in the results of the analysis.

P2. The construction of the abstract model should be fully automatic, since the original model is likely to be unintelligible due to size.

P3. The method should be generic in order to be applicable to as wide a range of CAS models as possible.

P4. The abstract model should preserve user-defined observables of the original system. For instance, it should be possible to fully recover the dynamics of selected variables of the original model.

The techniques discussed here revolve around the notion of *differential equivalence*, an equivalence relation over the variables of a dynamical system that induces a reduced model where each macro-variable represents the aggregate dynamics of an equivalence class. Two distinct flavours have been provided in [16]. *Forward differential equivalence* (FDE) is such that a macro-variable describes the sum of the variables of an equivalence class. For instance, consider:

$$\dot{x}_1 = -x_1, \qquad \dot{x}_2 = k_1 \cdot x_1 - x_2, \qquad \dot{x}_3 = k_2 \cdot x_1 - x_3, \tag{3}$$

where k_1 and k_2 are constants and the 'dot' operator denotes the derivative operator (with respect to time). Then, $\{\{x_1\},\{x_2,x_3\}\}$ is an FDE because

$$\dot{x}_1 = -x_1, \qquad (x_2 \dot{+} x_3) = \dot{x}_2 + \dot{x}_3 = (k_1 + k_2) \cdot x_1 - (x_2 + x_3). \tag{4}$$

By the change of variable $y = x_2 + x_3$, this is equivalent to writing

$$\dot{x}_1 = -x_1 \qquad \dot{y} = (k_1 + k_2) \cdot x_1 - y.$$

In this quotient model, whenever the *initial condition* (at time point 0) satisfies $y(0) = x_2(0) + x_3(0)$ we get that $y(t) = x_2(t) + x_3(t)$ at all time points t.

Backward differential equivalence (BDE) equates variables that have the same solutions at all time points. In (3), $\{\{x_1\},\{x_2,x_3\}\}$ is also a BDE provided that $k_1 = k_2$. In this case, we obtain a quotient ODE by removing either equation between x_2 and x_3, say x_3, and rewriting every occurrence of x_3 as x_2:

$$\dot{x}_1 = -x_1 \qquad \dot{x}_2 = k_1 \cdot x_1 - x_2.$$

In [16], BDE is shown to be a notion that corresponds to language-specific notions of equivalence developed in the past. Among these are the the exact versions of fluid lumpability [120] and the differential bisimulation for the process algebra PEPA of [92], as well as the backward bisimulation for chemical reaction networks of [64]. These have all been discussed in Deliverable D3.2. In addition, it can be shown that the aggregation method presented in [61] for the class of so-called nested automata model is related to BDE, formally when the rate functions for the interactions can be expressed with the IDOL language of [16]. Unlike the aforementioned notions, IDOL is domain agnostic because it essentially corresponds to a fragment of nonlinear ODEs that

includes rationals and threshold-like functions such as minima and maxima. As a consequence, for instance it permits the specification of non-mass-action kinetics such as Hill's which were instead forbidden in [64].

Both FDE and BDE satisfy **P1** because the relationship between the original model and the abstract one is exact; there is, however, loss of information when FDE is applied because the individual traces of the members of an equivalence class may not be recovered in general. Differential equivalences are closely related to the notion of exact ODE lumpability, very well understood in the chemistry literature (e.g., [119, 107, 100]). However this approach lacks of an automatic way of identifying lumping schemes (e.g., [121]). To cope with this, i.e., to satisfy **P2**, restrictions are imposed to be able to develop minimisation algorithms.

Symbolic minimisation. In [16] each ODE variable is treated explicitly as a real function and a differential equivalence is encoded in a logical formula over ODE variables. Thus, checking whether a candidate partition is BDE/FDE can be done *symbolically* using an encoding into satisfiability modulo theories (SMT) [45]. In fact, differential equivalences belong to the quantifier-free fragment of first-order logic. It is possible to restrict the admissible ODE systems to those for which an SMT solver for nonlinear real arithmetic — e.g., Z3 [77] — is a decision procedure. This can be done by, roughly speaking, excluding trigonometric functions (somewhat satisfying **P3**).

Let us consider the example (3), assuming $k_1 = k_2 = 1$. The condition for $\{\{x_1\},\{x_2,x_3\}\}$ to be a BDE can be shown to correspond to requiring that related variables with equal assignments always have equal derivatives. This can be encoded in a logical formula ϕ thus:

$$\phi := x_2 = x_3 \Rightarrow k_1 \cdot x_1 - x_2 = k_2 \cdot x_1 - x_3.$$

The SMT check $sat(\neg\phi)$ looks for an assignment of the variables x_1, x_2, and x_3 for which $\neg\phi$ holds. Thus, the partition is a BDE if and only if the procedure returns "unsat" (as is obviously the case in this example). More interestingly, it is possible to exploit the ability of the solver to return a *witness* in case of satisfiability. This can be interpreted as a counterexample that distinguishes variables originally supposed to be equivalent. For instance, an SMT check for the candidate BDE partition $\{\{x_1,x_2,x_3\}\}$ might return the witness $(x_1 = 1, x_2 = 1, x_3 = 1)$, which yields derivatives that are not equivalent $(\dot{x}_1 = -1, \dot{x}_2 = 0, \dot{x}_3 = 0)$. This suggests the implementation of an algorithm that *splits* the partition in such a way as to preserve the equalities in the witness. That is, at the next iteration the candidate BDE partition would be $\{\{x_1\},\{x_2,x_3\}\}$. It turns out that such an algorithm does iteratively compute the largest BDE that refines a given initial partition of ODE variables. This algorithm also meets **P4**: indeed each variable that should be treated as an *observable* can be put in a singleton initial block.

Syntax-driven minimisation. A more efficient minimisation algorithm can be provided for ODEs with derivatives that are multivariate polynomials of degree at most two [14]. This covers the ubiquitous linear systems as well as chemical reaction networks, at the basis of the aforementioned dynamic models in systems biology. At the basis of this approach is a finitary representation of an ODE system as a so-called *reaction network* (RN), consisting of species/variables interacting by means of reactions parameterised by a real value. On this representation two bisimulation equivalences, the forward (FB) and backward (BB) RN bisimulations, are related to FDE and BDE, respectively. This makes such bisimulations similar in spirit to quantitative equivalences on labelled transition systems, e.g., Larsen and Skou's probabilistic bisimulation [98]. In particular, the computation of the largest RN bisimulations that refine a given partition can be computed using an appropriate variant of Paige and Tarjan's famous algorithm [108]. In [14] a partition refinement algorithm is developed along the lines of efficient analogues for Markov chain lumping such as [79] and [122], and for probabilistic transition systems [43]. This computes the largest FB/BB refining a given partition of variables in $O(mn\log n)$ time, where m is the number of monomials in the ODE system and n is the number of variables. This algorithm is a significant improvement over a less sophisticated approach presented in [64] (and discussed in Deliverable D3.2) for chemical reaction networks.

Both the symbolic and the syntax-driven miminisation techniques are supported by a tool, *ERODE* [15], which is discussed in Deliverable 5.3 with further accompanying examples.

4 Towards Scalable Stochastic Variability Analysis

4.1 Background

Exploring the extension of variability analysis techniques as known from software product line (SPL) or product family engineering towards scalable stochastic variability analysis techniques suitable for CAS has been one of the research themes of WP3 throughout the months M7–M48 of the project, developed in the context of Task 3.3: *Relating Local and Global System Views with Variability Analysis*. After providing some background on SPLs and variability analysis, we briefly recall the main results from the previous reporting period, as addressed in Deliverable 3.2. Publications that appeared during this last reporting period, or that were not addressed in previous deliverables, and that are relevant to Task 3.3 are [7] concerning the synthesis of a (global) family model from a set of (local) product models, [3, 8, 9] concerning family-based model checking, [5, 6] concerning statistical model checking of dynamic SPLs with probabilistic behaviour, [1, 4] concerning the assessment of predictive services and variability-based decision support for bike-sharing systems and, finally, [24, 25] concerning variability-aware performance modelling.

4.1.1 Family-Based Analysis

An SPL is a family of products that can be distinguished (configured) according to a variability model defining their common (global) and variable (local) features. If feature combinations (i.e. product configurations) can change at run time, then we speak of *dynamic* SPLs (DSPLs). Analysis techniques for proving the correctness of behavioural SPL models are widely studied (cf. [117] for a survey). Since the number of configurable products in an SPL can be exponential in the number of features, enumerative *product-based* analysis of individual products quickly becomes infeasible. Therefore, dedicated *family-based* modelling and analysis techniques, dealing with entire SPLs at once using variability knowledge about valid feature combinations to deduce models and results for products, have been developed [89, 73, 76, 118, 47, 80]. Family-based behavioural modelling languages are usually based on superimposing multiple labelled transition systems (LTSs) representing products in a single, enriched LTS family model. Featured transition systems (FTSs) [73, 72, 71] are a popular example. In family-based analysis, once a property is verified for a family model one knows that the result also holds for any of its product models. This is in general more efficient than product-based analysis, in which every product thus has to be examined individually.

As anticipated in Deliverable 3.2, in [3] we contributed to this approach with the development of a modelling and analysis framework in which the family model is to be specified in a process-algebraic language, with a semantic interpretation as a modal transition system (MTS), together with an additional set of variability constraints. Properties to be verified (by means of efficient on-the-fly model checking) are to be formalised in a variability-aware variant of action-based CTL (called v-ACTL) that takes the modality of transitions into account. Given an MTS model of a product family, the framework supports both family-based analysis, based on results on the preservation of specific v-ACTL properties from an MTS to its set of product LTSs, as well as product-based analysis, upon the generation of all valid product LTSs from the MTS. Moreover, all these features are implemented in the variability model checker VMC (cf. `fmt.isti.cnr.it/vmc`) that also allows MTS/LTS visualisation, minimisation, etc., as we illustrated in Deliverable 3.2.

As explained in detail in Deliverable 3.2, in [52, 53, 55] we showed how the formal specification language `mCRL2` and its industrial-strength toolset (cf. `www.mcrl2.org`) can be exploited to model and analyse SPLs. In particular, the use of `mCRL2`'s parameterised data language to model and select valid product configurations, in the presence of feature attributes and quantitative constraints, and to model and check the behaviour of individually generated products, i.e. by means of *product-based* model checking, was illustrated. While we equipped the `mCRL2` models of product families with an FTS-like semantics, to perform *family-based* model-checking also the supporting logic (a variant of the first-order modal μ-calculus augmented with data) needs to be able to deal with the transitions of FTSs labelled with feature expressions.

4.1.2 Quantitative Analysis

As explained in detail in Deliverable 3.2, the family of *feature-oriented languages* `FLan` [51], `PFLan` [49], and `QFLan` [50] was developed during the project with the final aim of modelling and analysing DSPLs. A rich set of process-algebraic operators allows one to specify both the configuration and the behaviour of products of an SPL, while a constraint store allows one to specify all common constraints known from the variability models that come with SPLs as well as additional action constraints reminiscent of FTSs. Process execution is constrained by the store (e.g., to avoid introducing inconsistencies) but a process can also query the store (e.g., to resolve configuration options) or update the store (e.g., to add new features, even at run time). `PFLan` adds the possibility to equip actions (including those that install a feature, possibly at run time) with probability weights, which can represent uncertainty, failure rates, randomisation or simply relative preferences, resulting in *probabilistic* models of DSPLs in the form of DTMCs (after appropriate normalisation of the weights). An efficiently executable implementation in `Maude`, together with the distributed statistical model checker `MultiVeStA` [114] (developed during the first reporting period of the project and described in Deliverable 3.1) allows one to estimate the likelihood of specific configurations and behaviour of an SPL, and thus to measure non-functional aspects such as quality of service, reliability, or performance. `QFLan` adds the dynamic uninstallation and replacement of features and advanced *quantitative constraint modelling* options, allowing for more involved quantitative analyses requiring SMT solving. This was achieved by integrating an efficiently executable `Maude` implementation of `QFLan` with Microsoft's `Z3` [105] and with `MultiVeStA`.

As mentioned in Deliverable 3.2, in [42] we explored the possibility of applying *machine-learning* techniques to implement and evaluate predictive features of a bike-sharing system. We concentrated on predictive features capable of analysing the current state and historical data to provide the user with useful quantitative information, like the chances of finding a bike or an empty slot at a given docking station. The general goal was to evaluate feature combinations to help stakeholders decide which product of an SPL to deploy, making the best possible compromise between cost and usefulness.

Finally, previous work [95] discussed in Deliverable 3.2 presented the first family-based approach to performance analysis using a formalisation of UML Activity Diagrams (ADs). To capture performance properties, ADs are augmented with annotations (such as the duration to execute an activity of an activity node) and interpreted as continuous-time Markov chains, along the lines of established routes in model-driven software performance engineering. These *performance-annotated ADs* (PAADs) are integrated with SPL techniques to precisely capture variability aspects through a *delta-oriented* approach, where possibly many variants can be generated as a result of applying changes (i.e., *deltas*) to a *core* PAAD [113]. Here we briefly report on a significant improvement over [95] which tackles fundamental restrictions on the assumption of exponentially distributed service times and on a single-server semantics of the underlying queuing network model [24].

4.2 Objectives for the Reporting Period

We briefly recall from Deliverable 3.2 and from the Description of Work the objectives for this last reporting period concerning the work on scalable stochastic variability analysis.

- Synthesis of a compact (family) model of a large population built from smaller populations based on the commonalities and variability of single entities (product models) in their overall environment (SPL) and investigating the conditions under which properties can be preserved bottom-up as well as top-down, i.e. from local to global system views and vice versa.
- Development of a feature-oriented modal μ-calculus.
- Improving the `QFLan` implementation so that it scales better and evaluating its (improved) performance.
- Evaluating the expressivity and scalability of some of the modelling and verification frameworks for (family-based) behavioural variability analysis of (D)SPLs discussed in Deliverable 3.2 (e.g., those based on `QFLan`/`MultiVeStA` and FTSs/`mCRL2`).

- Applying the quantitative formal approaches to variability analysis for SPLs discussed in Deliverable 3.2 (e.g., statistical model checking of QFLan models and the exploitation of mature machine-learning techniques to assess prediction services for bike-sharing systems) in the context of smaller and larger scale CAS (e.g. in the context of smart city applications such as bike-sharing systems), allowing the evaluation of alternative designs or configurations (also at run time).

4.3 Achievements Concerning SPL Synthesis

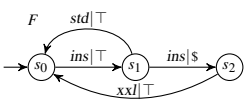
The CIF 3 toolset [54] targets the model-based engineering of supervisory controllers and supports such an engineering process by offering functionality for modelling, simulation, visualisation, synthesis, and code generation. In [7], we have used CIF 3 for feature-guided synthesis of SPLs, the first application of supervisory controller synthesis in SPL engineering. Such synthesis is a correct-by-construction approach to the development of a supervisor (or supervisory controller) capable of coordinating an assembly of (local) components into a (global) system that moreover functions correctly. Supervisory control theory [111] synthesises such a supervisory control model from models of system components and a given set of requirements. Moreover, the ensemble of components controlled by the supervisor satisfies a number of desirable properties, like the possibility to reach stable local states and the impossibility to globally disable events under local control.

More concretely, in [7] we showed how the CIF 3 toolset can automatically synthesise a single (family) model representing an automaton for each of the valid products of an SPL from (i) a rich variability model, (ii) a number of behavioural component models associated with the features of the SPL, and (iii) additional behavioural requirements like state invariants, event orderings, and guards on events (reminiscent of FTSs). The resulting CIF 3 model (if possible at all, otherwise CIF 3 reports that no such composition exists) is such that each initial state of the (global) system corresponds to a unique valid (local) product from the SPL as defined by the variability model, i.e. it satisfies all feature-related constraints as well as all behavioural requirements *by construction*. CIF 3 moreover allows the export of such models in a format accepted by the mCRL2 model checker, which can thus be used to verify behavioural properties expressed in the modal μ-calculus with data or in its feature-oriented variants of [8, 9]. It is important to underline that both CIF 3 and mCRL2 can be used off-the-shelf, meaning that no additional tools are required for which a level of trust needs to be established. Finally, we note that the explicit consideration of features as first-class citizens is a completely new way of using the CIF 3 toolset.

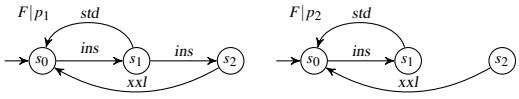
4.4 Achievements Concerning Family-Based Model Checking

Family-based model checking was proposed as a means to simultaneously verify multiple variants in a single run (cf. [117]). To make SPL models amenable to family-based model checking, feature-based variability was introduced in behavioural models. In particular, an FTS compactly represents multiple product behaviours in a single transition system by exploiting transitions guarded by so-called *feature expressions*.[5] A transition of a given product can be taken if the product fulfills the feature expression associated with the transition. Thus, an FTS F incorporates the behaviour of all eligible products, while the individual behaviour of a product p can be extracted as the LTS $F|p$.

Example 4.1 ([8, 9])**.** *Consider a product line P of (four) coffee machines, with independent features* $\{\$, €\}$ *representing the presence of a coin slot which accepts dollars or euros. The FTS F models its family behaviour. We see that P has actions for coin insertion (*ins*) and for pouring standard (*std*) or extra large (*xxl*) coffee. Each coffee machine accepts either dollars or euros. However, note that extra large coffee is exclusively available for two dollars.*



[5] A feature expression γ, a Boolean expression over a given set of features with f as typical element, can be interpreted as a set of products P_γ, viz. the products p for which the induced truth assignment (true for $\mathsf{f} \in p$, false for $\mathsf{f} \notin p$) validates γ. The Boolean constants verum ($\top$) and falsum ($\bot$) denote the feature expressions that are always true and always false, respectively.



The LTSs $F|p_1$ and $F|p_2$ model the behaviour of the products $p_1 = \{\$\}$ and $p_2 = \{\in\}$ of P. Note that $F|p_2$ lacks the transition from s_1 to s_2 which requires feature \$. □

The modal μ-calculus μL [96], which subsumes LTL and CTL, is used to express and model check properties interpreted over LTSs. During this reporting period, we proposed in [8] two variants of μL to reason about FTSs, by explicitly incorporating feature expressions into the logics, thus allowing operators to single out transitions and behaviour restricted to specific products and subfamilies. We provided semantics for these variants and related the logics to each other. Given that LTL and CTL are strict, partly overlapping subsets of μL, each of the feature-oriented variants we introduced can thus express properties that the approaches based exclusively on LTL and CTL cannot (cf. [69] for examples of such properties).

One of these variants, the logic μL_f, takes families (i.e. sets of products) as point of view. To this end, the standard modalities $\langle a \rangle$ and $[a]$ of μL are replaced by modalities $\langle a|\gamma \rangle$ and $[a|\gamma]$, respectively, where γ is a feature expression. Intuitively, $\langle a|\gamma \rangle \varphi_f$ holds for a family P with respect to an FTS F in a state s, if all products in P satisfy the feature expression γ and there is an a-transition, *shared among all products in P*, that leads to a state where φ_f holds for P (i.e. the products in P can collectively execute a). The difference with the corresponding modality in μL is less striking for $[a|\gamma]\,\varphi_f$, which holds in a state of F for a set of products P, if for each subset P' of P for which an a-transition is possible, φ_f holds for P' in the target state of that a-transition. Hence, $[a|\gamma]\,\varphi_f$ is trivially fulfilled for P if no product of P satisfies the feature expression γ. Otherwise, φ_f is checked against subsets P' of P cut out by γ and the feature expressions decorating the a-transitions of F.

Contrary to the modalities in μL, the modal operators of μL_f are not each other's dual, as shown in [8]. However, in [8] we proved that model checking a μL_f-formula for an individual product against an FTS reduces to model checking its *projection* against the LTS of the product. More precisely, given a formula $\varphi_f \in \mu L_f$ and a product $p \in \mathscr{P}$, a μL-formula $pr(\varphi_f, p)$ for φ_f with respect to p is obtained from φ_f, by replacing subformulae $\langle a|\gamma \rangle \psi_f$ by $\bot$ and $[a|\gamma]\,\psi_f$ by $\top$, respectively, in case $p \notin Q_\gamma$, while omitting γ otherwise. Formally, we proved the following.

Theorem 4.1 ([8])**.** *Let F be an FTS, and let $\mathscr{P}$ be a set of products.*

1. *For each formula $\varphi_f \in \mu L_f$, state $s \in S$, and individual product $p \in \mathscr{P}$:*
   $s, \{p\} \models_F \varphi_f \iff s \models_{F|p} pr(\varphi_f, p)$
2. *For negation-free formula $\varphi_f \in \mu L_f$, state $s \in S$, and product family $P \subseteq \mathscr{P}$:*
   $s, P \models_F \varphi_f \implies \forall p \in P\colon s \models_{F|p} pr(\varphi_f, p)$ □

During this reporting period, we subsequently showed in [9] how to effectively perform family-based model checking for μL_f by exploiting the mCRL2 toolset *as-is*, i.e. avoiding the implementation of a dedicated SPL-oriented verifier. We first defined a family-based partitioning procedure for μL_f that allowed us to apply the results from [8], after which we showed how to solve the resulting family-based model-checking problem via an embedding of μL_f into mCRL2's modal μ-calculus with data. More precisely, we first defined a family-based partitioning algorithm for computing a partitioning $(P_\oplus, P_\ominus)$ of a product family P such that each product in $P_\oplus$ satisfies the μL_f-formula φ_f, whereas each product in $P_\ominus$ fails φ_f. Clearly, repeatedly splitting families into subfamilies may lead to an exponential blow-up, in the worst case ultimately yielding a product-based analysis. However, in the SPL setting, an obvious strategy to partition a family of products P is to split it along a feature f. In general, the order of subsequent features f will influence the number of split-ups needed, but fortunately candidate features for splitting may be distilled from the structure of the system and from specific domain knowledge. Next, we defined how to translate (i) a μL_f-formula into a μL_{FO}-formula and (ii) an FTS into an LTS with parameterised actions, thus allowing family-based model checking with off-the-shelf tools that accept the logic from [86, 88] (of which μL_{FO} is a fragment), like the mCRL2 toolset.

4.4.1 Evaluation

We evaluated the family-based model-checking approach by verifying representative properties (cf. [71, 80]) over an mCRL2 specification of the minepump SPL benchmark family model [71, 94, 63, 80, 101]. This model

was first introduced in [73] as a reformulation of the configurable software system controlling a pump for mine drainage. Its purpose is to pump water out of a mine shaft, for which a controller operates a pump that may not start nor continue running in the presence of a dangerously high level of methane gas. Therefore, it communicates with a number of sensors measuring the water and methane levels. We considered the model as used in [71] that consists of 7 independent optional features for a total of $2^7 = 128$ variants. These features concern command types, methane detection, and water levels, abbreviated as *Ct*, *Cp*, *Ma*, *Mq*, *Lℓ*, *Ln*, and *Lh*.

The minepump model of [71] is distributed with the `ProVeLines` SPL toolset [75] (`projects.info.unamur.be/fts/provelines/`). We first manually translated the `fPROMELA`[6] model to a parameterised LTS encoded in `mCRL2`.[7] We then model checked twelve properties expressed in μL_f. The first six are μ-calculus versions of LTL properties of [71] (four of which are analysed also in [80]). The others are CTL-like properties. The final three, however, are feature-rich μL_f-formulae involving more than one feature modality in a single formula, which is a novelty that allows one to check different behaviour for different variants *at once*. Following the approach described above, the formulae were translated into μL_{FO} and model checked over the `mCRL2` model representing a parameterised LTS. The properties, results, and run times are summarised in Table 1.[8]

For each of the properties, we provide its intuitive meaning, its specification in μL_f, and the result of model checking it (indicating also the number of products for which the result holds). This concerns the first three columns of Table 1.[9,10] In the remaining columns, we report the run times (in seconds) needed to verify the properties with `mCRL2`, both product-based (*one-by-one*, abbreviated to 'one') and family-based (*all-in-one*, abbreviated to 'all'). We immediately notice the improvement in run time resulting from using `mCRL2` for family-based model checking as opposed to product-based model checking. The average speed up over these 8 properties is approximately 31, ranging from a worst speed up of slightly less than 4 (property φ_{12}) to a best speed up of over 97 (property φ_3). The average speed up over the 12 properties from [9] is approximately 30.

4.5 Achievements Concerning Statistical Model Checking DSPLs with Probabilistic Behaviour

The probabilistic feature-oriented language `QFLan` is a rich process algebra whose operational behaviour interacts with a store of constraints and as such product configuration is neatly separated from product behaviour. `QFLan` allows one to model a family of products with probabilistic behaviour, and possibly subject to quantitative feature constraints. Moreover, `QFLan` also caters for the dynamic installation, removal, or replacement of features, i.e. `QFLan` is suitable for modelling DSPLs. The resulting probabilistic configurations and behaviour converge seamlessly in a semantics based on DTMCs, thus enabling quantitative analysis. During this reporting period, we developed novel multi-platform `Eclipse`-based tool support for `QFLan`, which includes state-of-the-art domain-specific language utilities based on the `Xtext` framework as well as analysis plug-ins to run statistical model checking analyses from `Eclipse`. In [5], we provided a comprehensive presentation of our approach to model and analyse DSPLs with the `QFLan` modelling language, while we presented its novel tool support for the first time in [6].

The `QFLan` tool framework eases the modelling and analysis task by providing an `Eclipse` editor for `QFLan` specifications as well as plug-ins for the analysis. The architecture of the `QFLan` tool framework is depicted in Fig. 10. It is organised in a GUI layer, devoted to modelling aspects, and a core layer, offering support for the analysis of `QFLan` specifications. The components of the GUI layer are shown in Fig. 11. The most notable one is a text editor with editing support typical of modern integrated development environments (auto-completion, syntax and error highlighting, etc.) developed within the `Xtext` framework (top centre of Fig. 11). The editor also offers support for the `MultiQuaTEx` query language (cf. Deliverable 3.1), used to express properties of `QFLan` specifications. In addition, the layer offers a number of views, including (i) a *console view* to display diagnostic information (bottom of Fig. 11), (ii) a *project explorer* to collect different `QFLan` specifications

[6] `fPROMELA` is a feature-oriented extension of `PROMELA`, the input language of the `SPIN` model checker (cf. `www.spinroot.com`).

[7] The `mCRL2` code is distributed with the `mCRL2` toolset (svn revision 14493). All experiments were run on a standard Macbook Pro.

[8] We display only 8 of the 12 properties reported in [9] (two from each category), maintaining the original numbering used in [9].

[9] For a compact presentation of formulae in Table 1 we allow regular expressions in the modalities as syntactic sugar, as in [87, 88].

[10] Standard μ-calculus formulae in μL can be seen as μL_f-formulae by adjoining the feature expression $\top$ to every modality, i.e. replacing each 'diamond' modality $\langle a \rangle$ by $\langle a|\top \rangle$ and each 'box' modality $[a]$ by $[a|\top]$.

Table 1: Minepump properties and model-checking results (true/false) and run times (in seconds) of both product-based (*one-by-one*) and family-based (*all-in-one*) verification with `mCRL2`

Φ	property in μL_f	result	one	all
φ_1	*Absence of deadlock* $[\texttt{true}^*]\langle\texttt{true}\rangle\top$	128/0	10.02	2.07
φ_3	*The controller cannot fairly receive each of the three message types* $\mu X.([\texttt{true}^*.commandMsg]X \vee [\texttt{true}^*.alarmMsg]X \vee [\texttt{true}^*.levelMsg]X)$	0/128	24.33	0.25
φ_5	*The system cannot be in a situation in which the pump runs indefinitely in the presence of methane* $[\texttt{true}^*]\,(([pumpStart.(\neg pumpStop)^*.methaneRise]\,\mu X.[R]X)$ $\wedge([methaneRise.(\neg methaneLower)^*.pumpStart]\,\mu X.[R]X))$ for $R = \neg(pumpStop + methaneLower)$	96/32	17.26	0.86
φ_6	*Assuming fairness (φ_3), the system cannot be in a situation in which the pump runs indefinitely in the presence of methane (φ_5)* $[\texttt{true}^*]\,(([pumpStart.(\neg pumpStop)^*.methaneRise]\,\Psi)$ $\wedge([methaneRise.(\neg methaneLower)^*.pumpStart]\,\Psi))$ for $\Psi = \mu X.([R^*.commandMsg]X \vee [R^*.alarmMsg]X \vee [R^*.levelMsg]X)$ and $R = \neg(pumpStop + methaneLower)$	112/16	27.32	3.67
φ_7	*The controller can always eventually receive/read a message, i.e. it can return to its initial state from any state* $[\texttt{true}^*]\,\langle\texttt{true}^*.receiveMsg\rangle\top$	128/0	18.36	2.40
φ_9	*Invariantly, when the level of methane rises, it inevitably decreases* $[\texttt{true}^*.methaneRise]\;\mu X.[\neg methaneLower]X \wedge \langle\texttt{true}\rangle\top$	0/128	20.47	0.21
φ_{11}	*Products with feature Ct can always switch on the pump* $[\texttt{true}^*\,|\,Ct]\,\langle\texttt{true}^*.pumpStart\,|\,Ct\rangle\,\texttt{true}$	28/100	21.11	2.32
φ_{12}	*Products with features Ct, Ma, and Lh can start the pump upon a high water level, but products without feature Lh cannot* $[\texttt{true}^*\,|\,\top]\,((([highLevel\,|\,Ct \wedge Ma \wedge Lh]\,\langle\texttt{true}^*.pumpStart\,|\,\top\rangle\top) \wedge [pumpStart\,|\,\neg Lh]\,\bot)$	128/0	13.35	3.36

(left of Fig. 11), and (iii) a *plot view* to display analysis results (top right of Fig. 11). It does not require any installation process, and it is available together with usage instructions at `http://sysma.imtlucca.it/tools/multivesta/qflan/`.

A notable novelty of the new `QFLan` tool framework is a reimplementation of the `QFLan` interpreter that considerably reduces simulation time. There are a number of reasons for this improvement. First, the simulator is optimised by reusing computations performed in previous steps whenever possible. For instance, we re-check the admissibility of an action only if the constraint store has been modified in a way that could affect its admissibility. Second, checking whether an action is admissible is tantamount to checking whether it violates any constraint, which can be established using either `Z3`, accessed via its JAVA APIs, or an ad hoc constraint solver that we developed and which is much more effective than `Z3` in this particular setting, mainly because in this way we do not have to export the current status to `Z3` every time the store is updated.

In [6], we evaluated the new tool by applying it to two case studies. First, we applied it to the bikes product line from [50, 5], which was also briefly described in Deliverable 3.2. This showed a considerable improvement in performance, reducing analysis time by two orders of magnitude (from 45 minutes to 15 seconds). Second, we evaluated our approach by verifying a number of representative properties over a QFLan specification of the elevator SPL benchmark model [110, 72, 41, 74, 70, 65]. We considered more floors and more features than in any of these papers, resulting in a family of more than 2^{10} different products. The results showed that our approach scales well.

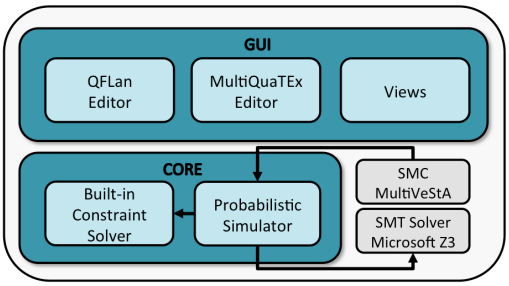


Figure 10: Architecture of the QFLan tool framework

4.6 Achievements Concerning the Assessment of Predictive Services and Variability-Based Decision Support for Bike-Sharing Systems

A bike-sharing system is a sustainable smart transportation system with a positive impact on urban mobility. Its design is multi-faceted and complex because of its many components, like bikes and docking stations, and its dynamicity, due to the actual cyclists. The latter form an intrinsic part of a bike-sharing system and their behavioural patterns impact the collective usability and performance of the system. One possibility to improve customer satisfaction for a bike-sharing system is to provide information on the status of the stations at run time. Indeed, most of the available systems currently provide such information in terms of the number of bikes parked in each docking station through web services. However, if one is looking for a bike it is of course nice to know that a station is currently empty, but it would be more valuable to know how the situation will evolve in the next few minutes and whether there is a chance that a bike is going to arrive; and likewise for full stations when one is looking for an empty slot to park one's bike. In [42], we envisaged services providing such predictions based on the expected movements of bikes towards and from a given station.

In [1], we provided an in-depth experimental assessment of the feasibility of such predictive services. We used different machine-learning models to implement and assess different predictive services, comparing several configurations and alternative means to aggregate historical data with different predictive performances and implementation costs. We performed our experiments based on real-world usage data covering all bike rentals in the *CicloPi* bike-sharing system of Pisa across two years (comprising over 280.000 entries). One of the outcomes was that there appears to be a consistent group of users whose usage behaviour is extremely predictable (with a combined performance measure, the so-called F1-score, of over 0.95 on a scale from 0 to 1) which is likely to correspond to people using *CicloPi* for commuting daily to work/school.

The advantage of a machine-learning approach is twofold. On the one hand, it provides a powerful and general means to realise a wide range of predictive services for which there exist sufficient (and significant) historical data. On the other hand, trained machine-learning models are provided with a measure of predictive performance that can be used as a metric to assess the cost-performance trade-off of the service. This provides a way to analyse the run-time behaviour of different variants of the predictive services before actually putting them into operation. The obtained results can thus serve vendors and administrators of bike-sharing systems that plan to deploy the services in a specific bike-sharing system.

In [4], we subsequently exploited the SPL paradigm to address the variability of the (predictive) services that the managers of a bike-sharing system can offer their clients. To this aim, we modelled a family of smart bike-sharing services, including an enriched variability model with quantitative attributes related to the services' cost and customer satisfaction, based on the aforementioned performance measure obtained from applying several machine-learning techniques to data logs from actual system usage. Consequently, it was shown how a dedicated tool can be used to perform multi-objective optimisation in order to assist the maintainers of the system in choosing the optimal configuration, i.e. minimising the costs while maximising customer satisfaction.

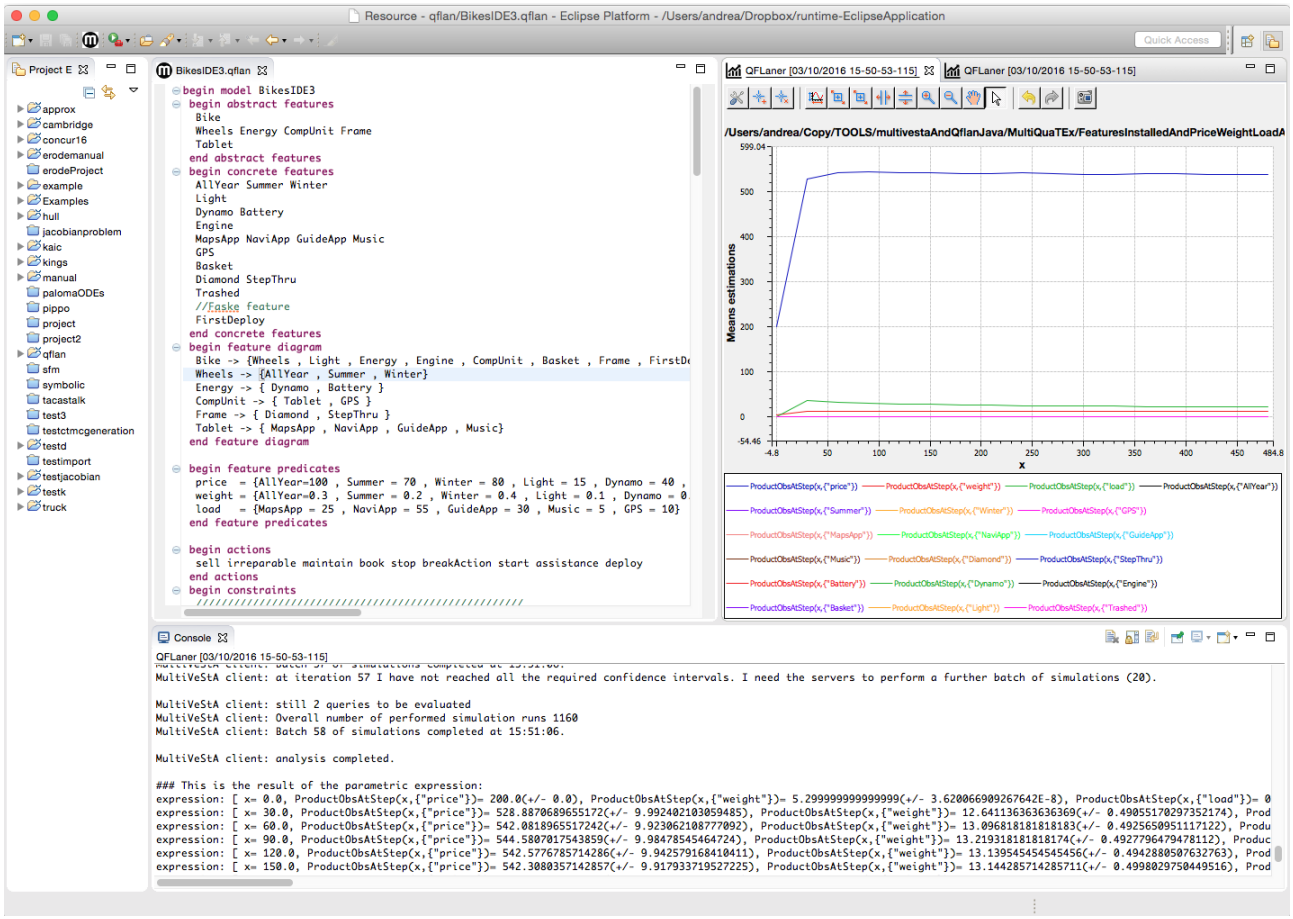


Figure 11: Screenshot of the QFLan tool framework

4.7 Achievements Concerning Variability-Aware Performance Modelling

There are two main benefits in using queuing networks (QNs) for model-based software performance prediction. First, there is a rather direct association between the constituent elements of a QN and the components of a software system. Indeed, QNs essentially capture the effects of *clients* contending for *resources*. Depending on the specific scenario, a QN client may be a user or an application; a QN *service station* may either represent a software device (e.g., a web server) or a hardware resource (e.g., CPU or disk); the number of servers at each station can be related to the degree of parallelism, e.g., the size of a thread pool or the number of CPU cores; the *routing probabilities*, which describe how a client visits another station upon service, can capture the operational profile of the workload. The second advantage in using QNs is that a number of well-known efficient techniques are available that cope well with state-space explosion such as mean value analysis.

Performance-annotated activity diagrams (PAADs) are a recent formalisation of a subset of UML activity diagrams carrying performance information which are amenable to family-based analysis through a symbolic solution of an underlying QN [95]. However, the analysis was restricted to assuming single-class queuing centres and exponentially distributed service times. Thus, family-based techniques cannot be used if the modeller wants to capture increasing degrees of parallelism/concurrency and general (non-exponential) service-time distributions. In this case, one cannot but resort to expensive simulations. To tackle this issue, in [24] a two-step solution is proposed that departs from the traditional solution techniques in two ways. First, the queueing model supports service times defined by Coxian distributions. These can be informally considered as a "composition" of exponential *stages* that can approximate any given probability distribution arbitrarily closely while still keeping the whole network representable as a CTMC [116]. This is instrumental for our second step, where we consider an approximation based on a fluid limit, where the underlying system of ODEs depends only on

the queueing network's topology and not on the number of CTMC states, which instead grow combinatorially with the number of stations and clients in the QN.

Following established product-line terminology, a PAAD consists of a kernel model, the *core*, together with a set of deltas. A delta is a list of basic operations such as the addition, removal, or modification of the performance annotations of nodes and edges of the PAAD. Overall, a delta transforms the core model into a new model *variant* [113]. We build a single super-PAAD, the so-called *150% model*, from which each variant can be obtained by computing an appropriate projection.

The main contribution of [24] is to show that the 150% model leads to a closed form *symbolic solution*, called the *family-based solution*, for the steady-state performance indices of every variant, for example its throughput. An essential contribution to finding such a symbolic expression, that can be efficiently evaluated, originates from the idea to replace the stochastic semantics of a PAAD, which grows combinatorially with the number of elements in the involved populations, by an ODE semantics as a compact system of ordinary differential equations.

The symbolic solution consists of rational expressions where the model parameters appear as unevaluated symbols whenever they are changed by at least one delta. The performance indices of any variant can be simply obtained by replacing these symbols with the concrete values related to that variant. A substantial experimental study shows that the family-based approach allows for the efficient treatment of large parameter spaces, because the symbolic expression is built only once for the whole family while its evaluation with concrete parameters is typically considerably faster than numerically solving a single variant in isolation.

5 Conclusions

In this deliverable we reported on the progress made concerning the objectives of all tasks of Work Package 3, namely Task 3.1, Task 3.2 and Task 3.3 during the final reporting period of the project, and in addition on progress made in Task 3.1 during the second reporting period that was not reported in earlier deliverables. In particular we addressed the development of spatio-temporal and scalable model checking techniques for the efficient analysis of CAS, model reduction techniques, both for the aggregation of CTMC population models and for those based on Ordinary Differential Equations, and the development, extension and application of techniques to deal with family-based variability analysis in the context of collective adaptive systems. There have not been any significant deviations from the objectives for this work package as foreseen in the description of work and in the previous deliverables and internal reports. A number of results have been shown to be much more versatile than expected, in particular the work on spatial and (stochastic) spatio-temporal model checking has shown to be successfully applicable also in research areas far outside that of collective adaptive systems such as developmental biology, biochemistry and even medical imaging. This holds also for some of the work on model reduction techniques for ODEs and might be expected also for fluid model checking and mean field model checking techniques. Some of these techniques could be expected to become a base for some future exploitable innovation.

Main achievements. Below we briefly summarise the main achievements addressed in the current report:

- Development of sophisticated global fluid model checking algorithms for the analysis of properties of an individual object in the context of a large continuous time population model and for global reachability properties (exploiting Linear Noise Approximation). The former algorithms have also been extended to deal with various types of rewards.

- Development of an open-source prototype on-the-fly mean field model checker (FlyFast), based on the theory and algorithms developed in WP3, and its application on various case studies.

- Development and extension of spatio-temporal model checking and monitoring algorithms and their open-source prototype implementation and application on case studies in the context of CAS (bike sharing), developmental biology and biochemistry. In particular, in the last reporting period spatial logic

operators with forms of distance metrics have been developed, as well as collective operators to reason about sets of points in space instead of reasoning about individual points. Also the robustness of properties has been addressed together with the combination of spatio-temporal logics with statistical model checking.

- Unexpected promising results of the application of spatial model checking (`topochecker`) in a completely different area of application, namely the analysis of medical images, where it was used to distinguish various "regions of interest" such as tumour or oedema tissue in the brain. Special spatial operators for statistical texture analysis and distance transform were added for this purpose and combined with the spatial operators from SLCS.
- Development of a new, high-level property-driven approach to model aggregation for CTMCs that identifies macro-states exploiting smoothed model checking.
- Full characterisation of equivalence relations (i.e. forward and backward differential equivalence) for ODEs identifying well-behaving, necessary and sufficient conditions for aggregation and the design and the development of efficient, fully automatic and customisable minimisation algorithms, exploiting symbolic and syntax-driven techniques.
- A co-algebraic characterisation of bisimulation of Labelled State-to-Function Transition Systems. This semantics framework has also been used for the definition of the operational semantics of the CARMA language.
- The first application of supervisory controller synthesis in SPL engineering.
- The definition of an off-the-shelf family-based model-checking procedure for verifying modal μ-calculus properties of SPLs (in fact, of any configurable system).
- The development of an Eclipse-based QFLan tool framework for modelling the probabilistic behaviour of DSPLs and efficiently running statistical model checking analyses.

Relationship to other work packages.

WP1 Emergent Behaviour and Adaptivity. Spatio-temporal model checking offers a complementary approach to the a more analytical oriented methods for the identification of emergent behaviour that evolves in a spatial context that were pursued in WP1. However, recent results on the accuracy of mean field approximation developed in WP1 is also of direct relevance to fluid model checking and mean field model checking developed in WP3. There are also links between the work on controller synthesis in software product lines and the work on control in WP1.

WP2 Collective Adaptive Behaviour in Space. One of the ideas that has been pursued in the work on spatial verification in WP3 is to develop a spatial logic with basic spatial operators that is general enough to be applied in the diverse spatial settings that were identified in the context of WP2. Furthermore, metrics and distance functions have been added in an orthogonal way both in the spatio-temporal logics STLCS and SSTL, providing further spatial richness.

WP4 Language and Design Methodology. In WP3 the foundations have been developed for a variety of scalable verification techniques of which a number have been implemented as prototype tools. CARMA models can be used to generate stochastic or deterministic simulation traces suitable for spatial model checking in much a similar way as spatio-temporal model checking was applied on simulation traces from the MRP bike sharing model discussed in Section 2. Furthermore, part of a design pathway is illustrated in the final deliverable of WP4 that shows how continuous time population models of bike sharing, specified in CARMA, can be transformed and analysed using the on-the-fly mean field model checker FlyFast developed in WP3.

WP5 Model Validation and Tool Support. The foundations of a number of open-source prototype tools have been developed in WP3 and the tools are available for use for model validation. These tools, which include the FlyFast on-the-fly mean field model checker, the spatio-temporal model checker `topochecker`, the spatial signal temporal monitoring tool jSSTL and the ODE-based minimisation tool ERODE are presented on the tool-page of the QUANTICOL web-site and described, providing detailed examples of their use, in the final deliverable of WP5.

Foresight. All three tasks in Work Package 3 have made considerable contributions to the theoretical foundations of scalable and spatial formal analysis methods, leading to many novel verification techniques and tools for Collective Adaptive Systems and beyond. Mean field and fluid approximation methods have played a key role in that development. From a model checking perspective those methods made it possible to overcome the state space explosion problem when addressing the verification of properties of large scale CAS. From a model reduction perspective such methods led to novel notions of behavioural equivalence in an ODE setting, bringing in reach of verification many models that were previously intractable. The same can be said of mean field and fluid model checking. So far, only a selection of the theoretical contributions have found their way into open source verification tools, due to the limited resources and time, however, there is much interest, both from industry and from the research community, to continue these activities. Besides providing further tools, there are many interesting avenues of future research from this work. For example, the work on differential equivalences offers possibilities for symbolic computation. Such computations play a key role when considering models with one or more parameter variables in order to find, for instance, equivalences that hold under any possible assignment of such variables or for which assignments can be synthesised in such a way that a candidate partition is a differential equivalence. In the context of fluid model checking stochastic approximation is exploited to efficiently check time-bounded *global* reachability properties of Markov population models making use of Linear Noise Approximation. This opens the way to even more sophisticated model checking techniques and tools for a large class of population models.

Where mean field and fluid approximation are less appropriate as analysis techniques, for example when models involve parameter values that are *uncertain*, novel simulation-based model checking techniques have been developed (such as *smoothed model checking*) exploiting properties of Gaussian processes. This method is, up to now, the most efficient approach for parametrised verification able to address 5-6 parameters at once. Uncertainty in parameter values is a frequent phenomenon in the performance analysis of systems and it is a very active field of research where much progress can be expected in the near future.

Spatial and spatio-temporal model checking and monitoring have also opened up a new line of research with applications ranging from CAS to developmental biology, biochemistry and even to unexpected fields such as medical imaging. Logic-based approaches are extremely versatile and general, so it can be expected that ever more areas of application will be encountered. Also from a theoretical perspective there are still many open questions, in particular the quest for the right combination of basic spatial and spatio-temporal operators that form an optimal compromise between efficient verification, expressivity and usability. There are very many options for such operators and only a small fraction of combinations have been fully explored so far. Moreover, extending the logics to reason about *regions* of space and their relationships, instead of about individual points, would be very interesting for applications in, for example, Computer Vision and Artificial Intelligence. Also their combination and further integration with mean-field based techniques is a very interesting and challenging area of research that is far from being fully explored. Another area of interest is the study of *probabilistic* logics for space (and their related probabilistic models). As for the application of spatio-temporal logics in medical imaging some very promising pilot studies have been performed that could have a strong transformational impact on how medical image analysis will be performed in the future.

Mean field and fluid approximation techniques have also been shown to be extremely useful and promising in the area of variability-aware performance modelling, where they have been used to find symbolic solutions that allow for the efficient treatment of complete families of large scale system models at once. Also in the absence of large populations of objects, variability in software product lines poses questions of scalability by itself. To address such problems, a family-based model-checking procedure for verifying μ-calculus properties

of SPLs, and similar systems with variability, has been defined using off-the-shelf verifiers. However, the efficiency of *computing* a partitioning of an SPL from which one can read which products satisfy which formula strongly depends on the adopted strategy for splitting SPLs and may well constitute a bottleneck in practice. Hence it is important to find heuristics to guide this splitting. One possibility may be to deduce an effective strategy from the lattice of product families that can be obtained by exploring the FTS model and keeping track of (the largest) product families that are capable of reaching states. This lattice may even allow for determining a proper partitioning *a priori*. Another potentially promising direction is to split product families using information that is obtained from counterexamples. Finally, one might make use of the theory of Galois connections to establish suitable abstractions of models prior to model checking.

6 Acknowledgements

We would like to thank the Advisory Board of the QUANTICOL project for their valuable suggestions. Further more we thank Marco Bertini from PisaMo S.p.A., Pisa, and Marco Giuppone and Manuela Quario from Bicincittà, Turin, Italy, for fruitful discussions, collaboration and feedback on the results on Bike Sharing obtained in the context of the project.

References concerning publications of the Quanticol project for the third reporting period

[1] D. Bacciu, A. Carta, S. Gnesi, and L. Semini. "An Experience in Using Machine Learning for Short-term Predictions in Smart Transportation Systems". In: *J. Log. Algebr. Meth. Program.* 87 (2017), pp. 52–66. DOI: 10.1016/j.jlamp.2016.11.002.

[2] E. Bartocci, L. Bortolussi, D. Milios, L. Nenzi, and G. Sanguinetti. "Studying Emergent Behaviours in Morphogenesis Using Signal Spatio-Temporal Logic". In: *Hybrid Systems Biology - Revised Selected Papers of the 4th International Workshop on Hybrid Systems Biology (HSB'15)*. Ed. by A. Abate and D. Safránek. Vol. 9271. LNCS. Springer, 2015, pp. 156–172. DOI: 10.1007/978-3-319-26916-0_9.

[3] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. "Modelling and analysing variability in product families: Model checking of modal transition systems". In: *J. Log. Algebr. Meth. Program.* 85.2 (2016), pp. 287–315. DOI: 10.1016/j.jlamp.2015.11.006.

[4] M. H. ter Beek, A. Fantechi, S. Gnesi, and L. Semini. "Variability-Based Design of Services for Smart Transportation Systems". In: *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications (ISoLA'16)*. Ed. by T. Margaria and B. Steffen. Vol. 9953. LNCS. Springer, 2016, pp. 465–481. DOI: 10.1007/978-3-319-47169-3_38.

[5] M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. "Statistical Model Checking for Product Lines". In: *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*. Ed. by T. Margaria and B. Steffen. Vol. 9952. LNCS. Springer, 2016, pp. 114–133. DOI: 10.1007/978-3-319-47166-2_8.

[6] M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. "A Quantitative Approach to Model and Analyze Dynamic Software Product Lines Using Constraints and Statistical Model Checking". In: Submitted for publication (2017).

[7] M. H. ter Beek, M. A. Reniers, and E. P. de Vink. "Supervisory Controller Synthesis for Product Lines Using CIF 3". In: *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*. Ed. by T. Margaria and B. Steffen. Vol. 9952. LNCS. Springer, 2016, pp. 856–873. DOI: 10.1007/978-3-319-47166-2_59.

[8] M. H. ter Beek, E. P. de Vink, and T. A. C. Willemse. "Towards a Feature mu-Calculus Targeting SPL Verification". In: *Proceedings 7th International Workshop on Formal Methods and Analysis in Software Product Line Engineering (FMSPLE'16)*. Vol. 206. EPTCS. 2016, pp. 61–75. DOI: 10.4204/EPTCS.206.6.

[9] M. H. ter Beek, E. P. de Vink, and T. A. C. Willemse. "Family-Based Model Checking with mCRL2". In: *Proceedings of the 20th International Conference on Fundamental Approaches to Software Engineering (FASE'17)*. Ed. by M. Huisman and J. Rubin. Vol. 10202. LNCS. Springer, 2017, pp. 387–405. DOI: 10.1007/978-3-662-54494-5_23.

[10] G. Belmonte, V. Ciancia, D. Latella, and M. Massink. "From Collective Adaptive Systems to Human Centric Computation and Back: Spatial Model Checking for Medical Imaging". In: *Proceedings of the Workshop on FORmal methods for the quantitative Evaluation of Collective Adaptive SysTems (FORECAST'16)*. Ed. by M. H. ter Beek and M. Loreti. Vol. 217. EPTCS. 2016, pp. 81–92. DOI: 10.4204/EPTCS.217.10.

[11] L. Bortolussi, L. Cardelli, M. Kwiatkowska, and L. Laurenti. "Approximation of Probabilistic Reachability for Chemical Reaction Networks Using the Linear Noise Approximation". In: *Proceedings of the 13th International Conference on Quantitative Evaluation of Systems (QEST'16)*. Ed. by G. Agha and B. V. Houdt. Vol. 9826. LNCS. Springer, 2016, pp. 72–88. DOI: 10.1007/978-3-319-43425-4_5.

[12] L. Bortolussi and J. Hillston. "Efficient Checking of Individual Rewards Properties in Markov Population Models". In: *Proceedings 13th Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL'15)*. Ed. by N. Bertrand and M. Tribastone. Vol. 194. EPTCS. 2015, pp. 32–47. DOI: 10.4204/EPTCS.194.3.

[13] L. Bortolussi, D. Milios, and G. Sanguinetti. "Smoothed model checking for uncertain Continuous-Time Markov Chains". In: *Inf. Comput.* 247 (2016), pp. 235–253. DOI: 10.1016/j.ic.2016.01.004.

[14] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. "Efficient Syntax-Driven Lumping of Differential Equations". In: *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*. Ed. by M. Chechik and J.-F. Raskin. Vol. 9636. LNCS. Springer, 2016, pp. 93–111. DOI: 10.1007/978-3-662-49674-9_6.

[15] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. "ERODE: A Tool for the Evaluation and Reduction of Ordinary Differential Equations". In: *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'17)*. Ed. by A. Legay and T. Margaria. Vol. 10206. LNCS. To appear. Springer, 2017.

[16] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. "Symbolic computation of differential equivalences". In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'16)*. Ed. by R. Bodík and R. Majumdar. ACM, 2016, pp. 137–150. DOI: 10.1145/2837614.2837649.

[17] V. Ciancia, S. Gilmore, G. Grilletti, D. Latella, M. Loreti, and M. Massink. *On spatio-temporal model-checking of vehicular movement in transport systems – Preliminary version*. Tech. rep. TR-QC-02-2016. QUANTICOL, 2016.

[18] V. Ciancia, G. Grilletti, D. Latella, M. Loreti, and M. Massink. "An Experimental Spatio-Temporal Model Checker". In: *Software Engineering and Formal Methods - Revised Selected Papers of the SEFM 2015 Collocated Workshops: ATSE, HOFM, MoKMaSD, and VERY*SCART*. Ed. by D. Bianculli, R. Calinescu, and B. Rumpe. Vol. 9509. LNCS. Springer, 2015, pp. 297–311. DOI: 10.1007/978-3-662-49224-6_24.

[19] V. Ciancia, D. Latella, M. Loreti, and M. Massink. "Model Checking Spatial Logics for Closure Spaces". In: *Log. Meth. Comput. Sci.* 12.4 (2016). DOI: 10.2168/LMCS-12(4:2)2016.

[20] V. Ciancia, D. Latella, M. Loreti, and M. Massink. "Spatial Logic and Spatial Model Checking for Closure Spaces". In: *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems - Advanced Lectures of the 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'16)*. Ed. by M. Bernardo, R. De Nicola, and J. Hillston. Vol. 9700. LNCS. Springer, 2016, pp. 156–201. DOI: 10.1007/978-3-319-34096-8_6.

[21] V. Ciancia, D. Latella, and M. Massink. "On-the-Fly Mean-Field Model-Checking for Attribute-Based Coordination". In: *Proceedings of the 18th IFIP WG 6.1 International Conference on Coordination Models and Languages (COORDINATION'16), Held as Part of the 11th International Federated Conference on Distributed Computing Techniques (DisCoTec'16)*. Ed. by A. Lluch Lafuente and J. Proença. Vol. 9686. LNCS. Springer, 2016, pp. 67–83. DOI: 10.1007/978-3-319-39519-7_5.

[22] V. Ciancia, D. Latella, M. Massink, R. Paškauskas, and A. Vandin. "A Tool-Chain for Statistical Spatio-Temporal Model Checking of Bike Sharing Systems". In: *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*. Ed. by T. Margaria and B. Steffen. Vol. 9952. LNCS. 2016, pp. 657–673. DOI: 10.1007/978-3-319-47166-2_46.

[23] C. Feng, J. Hillston, and V. Galpin. "Automatic Moment-Closure Approximation of Spatially Distributed Collective Adaptive Systems". In: *ACM Trans. Model. Comput. Simul.* 26.4 (2016), 26:1–26:22. DOI: 10.1145/2883608. URL: 10.1145/2883608.

[24] M. Kowal, M. Tschaikowski, M. Tribastone, and I. Schaefer. "Scaling Size and Parameter Spaces in Variability-aware Software Performance Models". In: *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE'15)*. Ed. by M. B. Cohen, L. Grunske, and M. Whalen. Vol. 252. LNI. IEEE Computer Society, 2015, pp. 407–417. DOI: 10.1109/ASE.2015.16.

[25] M. Kowal, M. Tschaikowski, M. Tribastone, and I. Schaefer. "Scaling Size and Parameter Spaces in Variability-aware Software Performance Models". In: *Software Engineering 2016 - Fachtagung des GI-Fachbereichs Softwaretechnik*. Ed. by J. Knoop and U. Zdun. Vol. 252. LNI. GI, 2016, pp. 33–34. URL: http://subs.emis.de/LNI/Proceedings/Proceedings252/article33.html.

[26] D. Latella, M. Loreti, and M. Massink. "FlyFast: A Mean Field Model Checker". In: *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'17)*. Ed. by A. Legay and T. Margaria. Vol. 10206. LNCS. To appear. Springer, 2017.

[27] D. Latella and M. Massink. *Design and Optimisation of the FlyFast Front-end for Attribute-based Coordination. Preliminary Version*. TR TR-QC-01-2017. QUANTICOL, 2017.

[28] D. Latella, M. Massink, and E. P. de Vink. "Bisimulation of Labelled State-to-Function Transition Systems Coalgebraically". In: *Log. Meth. Comput. Sci.* 11.4 (2015). DOI: 10.2168/LMCS-11(4:16)2015.

[29] L. Luisa Vissat, M. Loreti, L. Nenzi, J. Hillston, and G. Marion. "Three-Valued Spatio-Temporal Logic: a further analysis on spatio-temporal properties of stochastic systems". In: *Submitted* (2017).

[30] M. Michaelides, D. Milios, J. Hillston, and G. Sanguinetti. "Property-Driven State-Space Coarsening for Continuous Time Markov Chains". In: *Proceedings of the 13th International Conference on Quantitative Evaluation of Systems (QEST'16)*. Ed. by G. Agha and B. V. Houdt. Vol. 9826. LNCS. Springer, 2016, pp. 3–18. DOI: 10.1007/978-3-319-43425-4_1.

[31] P. Piho and J. Hillston. "Stochastic and Spatial Equivalences for PALOMA". In: *Proceedings of the Workshop on FORmal methods for the quantitative Evaluation of Collective Adaptive SysTems (FORECAST'16)*. Ed. by M. H. ter Beek and M. Loreti. Vol. 217. EPTCS. 2016, pp. 69–80. DOI: 10.4204/EPTCS.217.9.

[32] M. Tribastone. "Challenges in Quantitative Abstractions for Collective Adaptive Systems". In: *Proceedings of the Workshop on FORmal methods for the quantitative Evaluation of Collective Adaptive SysTems (FORECAST'16)*. Ed. by M. H. ter Beek and M. Loreti. Vol. 217. EPTCS. 2016, pp. 62–68. DOI: 10.4204/EPTCS.217.8.

[33] A. Vandin and M. Tribastone. "Quantitative Abstractions for Collective Adaptive Systems". In: *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems - Advanced Lectures of the 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'16)*. Ed. by M. Bernardo, R. De Nicola, and J. Hillston. Vol. 9700. LNCS. Springer, 2016, pp. 202–232. DOI: 10.1007/978-3-319-34096-8_7.

References concerning publications of the Quanticol project for the second reporting period for Task 3.1

[34] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. "System design of stochastic models using robustness of temporal properties". In: *Theoret. Comput. Sci.* 587 (2015), pp. 3–25. DOI: 10.1016/j.tcs.2015.02.046. URL: 10.1016/j.tcs.2015.02.046.

[35] L. Bortolussi and J. Hillston. "Model checking single agent behaviours by fluid approximation". In: *Inf. Comput.* 242 (2015), pp. 183–226. DOI: 10.1016/j.ic.2015.03.002.

[36] V. Ciancia, D. Latella, M. Massink, and R. Paskauskas. "Exploring Spatio-temporal Properties of Bike-Sharing Systems". In: *Proceedings of the IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW'15)*. IEEE, 2015, pp. 74–79. DOI: 10.1109/SASOW.2015.17.

[37] D. Latella, M. Loreti, and M. Massink. "On-the-fly PCTL fast mean-field approximated model-checking for self-organising coordination". In: *Sci. Comput. Program.* 110 (2015), pp. 23–50. DOI: 10.1016/j.scico.2015.06.009.

[38] D. Latella, M. Loreti, and M. Massink. "On-the-fly Fluid Model Checking via Discrete Time Population Models". In: *Computer Performance Engineering — Proceedings of the 12th European Workshop on Computer Performance Engineering (EPEW'15)*. Ed. by M. Beltrán, W. Knottenbelt, and J. Bradley. Vol. 9272. LNCS. Springer, 2015, pp. 193–207. DOI: 10.1007/978-3-319-23267-6_13.

[39] M. Massink and R. Paškauskas. "Model-based Assessment of Aspects of User-satisfaction in Bicycle Sharing Systems". In: *Proceedings of the 18th IEEE International Conference on Intelligent Transportation Systems (ITSC'15)*. DOI: 10.1109/ITSC.2015.224. IEEE, 2015, pp. 1363–1370.

[40] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink. "Qualitative and Quantitative Monitoring of Spatio-Temporal Properties". In: *Proceedings of the 6th International Conference on Runtime Verification (RV'15)*. Ed. by E. Bartocci and R. Majumdar. Vol. 9333. LNCS. Springer, 2015, pp. 21–37. DOI: 10.1007/978-3-319-23820-3_2.

References

[41] S. Apel, A. von Rhein, P. Wendler, A. Größlinger, and D. Beyer. "Strategies for Product-line Verification: Case Studies and Experiments". In: *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*. IEEE, 2013, pp. 482–491. DOI: 10.1109/ICSE.2013.6606594.

[42] D. Bacciu, S. Gnesi, and L. Semini. "Using a Machine Learning Approach to Implement and Evaluate Product Line Features". In: *Proceedings 11th International Workshop on Automated Specification and Verification of Web Systems (WWV'15)*. Ed. by M. H. ter Beek and A. Lluch Lafuente. Vol. 188. EPTCS. 2015, pp. 75–83. DOI: 10.4204/EPTCS.188.8.

[43] C. Baier, B. Engelen, and M. E. Majster-Cederbaum. "Deciding Bisimilarity and Similarity for Probabilistic Processes". In: *J. Comput. Syst. Sci.* 60.1 (2000), pp. 187–231. DOI: 10.1006/jcss.1999.1683.

[44] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008. URL: http://mitpress.mit.edu/books/principles-model-checking.

[45] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. "Satisfiability Modulo Theories". In: *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. Ed. by A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. IOS Press, 2009. Chap. 12.

[46] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. "On the Robustness of Temporal Properties for Stochastic Models". In: *Proceedings 2nd International Workshop on Hybrid Systems Biology (HSB'13)*. Ed. by T. Dang and C. Piazza. Vol. 125. EPTCS. 2013, pp. 3–19. DOI: 10.4204/EPTCS.125.1.

[47] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. "Using FMC for Family-based Analysis of Software Product Lines". In: *Proceedings of the 19th International Software Product Line Conference (SPLC'15)*. ACM, 2015, pp. 432–439. DOI: 10.1145/2791060.2791118.

[48] M. H. ter Beek, S. Gnesi, D. Latella, and M. Massink. "Towards Automatic Decision Support for Bike-Sharing System Design". In: *Software Engineering and Formal Methods - Revised Selected Papers of the SEFM 2015 Collocated Workshops: ATSE, HOFM, MoKMaSD, and VERY*SCART*. Ed. by D. Bianculli, R. Calinescu, and B. Rumpe. Vol. 9509. LNCS. Springer, 2015, pp. 266–280. DOI: 10.1007/978-3-662-49224-6_22.

[49] M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. "Quantitative Analysis of Probabilistic Models of Software Product Lines with Statistical Model Checking". In: *Proceedings 6th International Workshop on Formal Methods for Software Product Line Engineering (FMSPLE'15)*. Ed. by J. M. Atlee and S. Gnesi. Vol. 182. EPTCS. 2015, pp. 56–70. DOI: 10.4204/EPTCS.182.5.

[50] M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. "Statistical Analysis of Probabilistic Models of Software Product Lines with Quantitative Constraints". In: *Proceedings of the 19th International Software Product Line Conference (SPLC'15)*. ACM, 2015, pp. 11–15. DOI: 10.1145/2791060.2791087.

[51] M. H. ter Beek, A. Lluch Lafuente, and M. Petrocchi. "Combining Declarative and Procedural Views in the Specification and Analysis of Product Families". In: *Proceedings of the 17th International Software Product Line Conference (SPLC'13)*. Vol. 2. ACM, 2013, pp. 10–17. DOI: 10.1145/2499777.2500722.

[52] M. H. ter Beek and E. P. de Vink. "Using mCRL2 for the Analysis of Software Product Lines". In: *Proceedings of the 2nd FME Workshop on Formal Methods in Software Engineering (FormaliSE'14)*. Ed. by S. Gnesi and N. Plat. IEEE, 2014, pp. 31–37. DOI: 10.1145/2593489.2593493.

[53] M. H. ter Beek and E. P. de Vink. "Towards Modular Verification of Software Product Lines with mCRL2". In: *Proceedings of the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'14)*. Ed. by T. Margaria and B. Steffen. Vol. 8802. LNCS. Springer, 2014, pp. 368–385. DOI: 10.1007/978-3-662-45234-9_26.

[54] D. A. van Beek, W. J. Fokkink, D. Hendriks, A. Hofkamp, J. Markovski, J. M. van de Mortel-Fronczak, and M. A. Reniers. "CIF 3: Model-Based Engineering of Supervisory Controllers". In: *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*. Ed. by E. Ábrahám and K. Havelund. Vol. 8413. LNCS. Springer, 2014, pp. 575–580. DOI: 10.1007/978-3-642-54862-8_48.

[55] T. Belder, M. H. ter Beek, and E. P. de Vink. "Coherent branching feature bisimulation". In: *Proceedings 6th International Workshop on Formal Methods for Software Product Line Engineering (FMSPLE'15)*. Ed. by J. M. Atlee and S. Gnesi. Vol. 182. EPTCS. 2015, pp. 14–30. DOI: 10.4204/EPTCS.182.2.

[56] J. van Benthem and G. Bezhanishvili. "Modal Logics of Space". In: *Handbook of Spatial Logics*. Ed. by M. Aiello, I. E. Pratt-Hartmann, and J. F. A. K. van Benthem. Springer, 2007, pp. 217–298. DOI: 10.1007/978-1-4020-5587-4_5.

[57] M. Bernardo, L. Donatiello, and R. Gorrieri. "A Formal Approach to the Integration of Performance Aspects in the Modeling and Analysis of Concurrent Systems". In: *Inf. Comput.* 144.2 (1998), pp. 83–154. DOI: 10.1006/inco.1998.2706. URL: http://dx.doi.org/10.1006/inco.1998.2706.

[58] L. Bortolussi and R. Lanciani. "Model Checking Markov Population Models by Central Limit Approximation". In: *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST'13)*. Ed. by K. Joshi, M. Siegle, M. Stoelinga, and P. R. D'Argenio. Vol. 8054. LNCS. Springer, 2013, pp. 123–138. DOI: 10.1007/978-3-642-40196-1_9.

[59] L. Bortolussi, R. De Nicola, V. Galpin, S. Gilmore, J. Hillston, D. Latella, M. Loreti, and M. Massink. "CARMA: Collective Adaptive Resource-sharing Markovian Agents". In: *Proceedings Thirteenth Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL'15)*. Ed. by N. Bertrand and M. Tribastone. Vol. 194. EPTCS. 2015, pp. 16–31. DOI: 10.4204/EPTCS.194.2.

[60] L. Bortolussi and J. Hillston. "Fluid Model Checking". In: *Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12)*. Ed. by M. Koutny and I. Ulidowski. Vol. 7454. LNCS. Springer, 2012, pp. 333–347. DOI: 10.1007/978-3-642-32940-1_24.

[61] L. Bortolussi, J. Hillston, and M. Tribastone. "Fluid Performability Analysis of Nested Automata Models". In: *ENTCS* 310 (2015), pp. 27–47. DOI: 10.1016/j.entcs.2014.12.011.

[62] L. Bortolussi, D. Milios, and G. Sanguinetti. "U-Check: model checking and parameter synthesis under uncertainty". In: *Proceedings of the 12th International Conference on Quantitative Evaluation of Systems (QEST'15)*. Ed. by J. Campos and B. R. Haverkort. Vol. 9259. LNCS. Springer, 2015, pp. 89–104. DOI: 10.1007/978-3-319-22264-6_6.

[63] J. Bürdek, M. Lochau, S. Bauregger, A. Holzer, A. von Rhein, S. Apel, and D. Beyer. "Facilitating Reuse in Multi-goal Test-Suite Generation for Software Product Lines". In: *Proceedings of the 18th International Conference on Fundamental Approaches to Software Engineering (FASE'15)*. Ed. by A. Egyed and I. Schaefer. Vol. 9033. LNCS. Springer, 2015, pp. 84–99. DOI: 10.1007/978-3-662-46675-9_6.

[64] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. "Forward and Backward Bisimulations for Chemical Reaction Networks". In: *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15)*. Ed. by L. Aceto and D. de Frutos-Escrig. Vol. 42. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 226–239. DOI: 10.4230/LIPIcs.CONCUR.2015.226.

[65] P. Chrszon, C. Dubslaff, S. Klüppelholz, and C. Baier. "Family-Based Modeling and Analysis for Probabilistic Systems – Featuring ProFeat". In: *Proceedings of the 19th International Conference on Fundamental Approaches to Software Engineering (FASE'16)*. Ed. by P. Stevens and A. Wąsowski. Vol. 9633. LNCS. Springer, 2016, pp. 287–304. DOI: 10.1007/978-3-662-49665-7_17.

[66] V. Ciancia. *topochecker - a topological model checker*. http://fmt.isti.cnr.it/topochecker – https://github.com/vincenzoml/topochecker. 2015.

[67] V. Ciancia, S. Gilmore, D. Latella, M. Loreti, and M. Massink. "Data Verification for Collective Adaptive Systems: Spatial Model-Checking of Vehicle Location Data". In: *Proceedings of the 8th IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW'14)*. IEEE Computer Society, 2014, pp. 32–37. DOI: 10.1109/SASOW.2014.16.

[68] V. Ciancia, D. Latella, M. Loreti, and M. Massink. "Specifying and verifying properties of space". In: *Proceedings of the 8th IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science (TCS'14)*. Ed. by J. Diaz, I. Lanese, and D. Sangiorgi. Vol. 8705. LNCS. Springer, 2014, pp. 222–235. DOI: 10.1007/978-3-662-44602-7_18.

[69] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999. URL: http://mitpress.mit.edu/books/model-checking.

[70] A. Classen, M. Cordy, P. Heymans, A. Legay, and P.-Y. Schobbens. "Formal semantics, modular specification, and symbolic verification of product-line behaviour". In: *Sci. Comput. Program.* 80.B (2014), pp. 416–439. DOI: 10.1145/2499777.2499781.

[71] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin. "Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking". In: *IEEE Trans. Softw. Eng.* 39.8 (2013), pp. 1069–1089. DOI: 10.1109/TSE.2012.86.

[72] A. Classen, P. Heymans, P.-Y. Schobbens, and A. Legay. "Symbolic model checking of software product lines". In: *Proceedings of the 33rd International Conference on Software Engineering (ICSE'11)*. ACM, 2011, pp. 321–330. DOI: 10.1145/1985793.1985838.

[73] A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay, and J.-F. Raskin. "Model Checking Lots of Systems: Efficient Verification of Temporal Properties in Software Product Lines". In: *Proceedings of the 32nd International Conference on Software Engineering (ICSE'10)*. ACM, 2010, pp. 335–344. DOI: 10.1145/1806799.1806850.

[74] M. Cordy, P.-Y. Schobbens, P. Heymans, and A. Legay. "Beyond Boolean Product-Line Model Checking: Dealing with Feature Attributes and Multi-features". In: *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*. IEEE, 2013, pp. 472–481. DOI: 10.1109/ICSE.2013.6606593.

[75] M. Cordy, A. Classen, P. Heymans, P.-Y. Schobbens, and A. Legay. "ProVeLines: a product line of verifiers for software product lines". In: *Proceedings of the 17th International Software Product Line Conference (SPLC'13)*. Vol. 2. ACM, 2013, pp. 141–146. DOI: 10.1145/2499777.2499781.

[76] F. Damiani and I. Schaefer. "Family-Based Analysis of Type Safety of Delta-Oriented Software Product Lines". In: *Proceedings of the 5th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'12)*. Ed. by T. Margaria and B. Steffen. Vol. 7609. LNCS. Springer, 2012, pp. 193–207. DOI: 10.1007/978-3-642-34026-0_15.

[77] L. De Moura and N. Bjørner. "Z3: An Efficient SMT Solver". In: *TACAS*. 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[78] R. De Nicola, D. Latella, M. Loreti, and M. Massink. "A uniform definition of stochastic process calculi". In: *ACM Comput. Surv.* 46.1 (2013), 5:1–5:35. DOI: 10.1145/2522968.2522973.

[79] S. Derisavi, H. Hermanns, and W. Sanders. "Optimal state-space lumping in Markov chains". In: *Information Processing Letters* 87.6 (2003), pp. 309–315. DOI: 10.1016/S0020-0190(03)00343-0.

[80] A. S. Dimovski, A. S. Al-Sibahi, C. Brabrand, and A. Wąsowski. "Family-Based Model Checking Without a Family-Based Model Checker". In: *Proceedings of the 22nd International SPIN Symposium on Model Checking of Software (SPIN'15)*. Ed. by B. Fischer and J. Geldenhuys. Vol. 9232. LNCS. Springer, 2015, pp. 282–299. DOI: 10.1007/978-3-319-23404-5_18.

[81] A. Donzé, T. Ferrère, and O. Maler. "Efficient Robust Monitoring for STL". In: *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13)*. Ed. by N. Sharygina and H. Veith. Vol. 8044. LNCS. Springer, 2013, pp. 264–279. DOI: 10.1007/978-3-642-39799-8_19.

[82] C. Feng and J. Hillston. "Speed-up of Stochastic Simulation of PCTMC Models by Statistical Model Reduction". In: *Proceedings of the 12th European Workshop on Computer Performance Engineering (EPEW'15)*. Vol. 9272. LNCS. Springer, 2015, pp. 291–305. DOI: 10.1007/978-3-319-23267-6_19.

[83] A. Galton. "A generalized topological view of motion in discrete space". In: *Theoret. Comput. Sci.* 305.1–3 (2003), pp. 111–134. DOI: 10.1016/S0304-3975(02)00701-6.

[84] G. J. Grevera. "Distance Transform Algorithms And Their Implementation And Evaluation". In: *Deformable Models - Biomedical and Clinical Applications*. Ed. by J. S. Suri and A. A. Farag. Topics in Biomedical Engineering. Springer, 2007, pp. 33–60. DOI: 10.1007/978-0-387-68413-0_2.

[85] G. Grilletti and V. Ciancia. *STLCS model checker*. https://github.com/cherosene/ctl_logic. 2014.

[86] J. F. Groote and R. Mateescu. "Verification of Temporal Properties of Processes in a Setting with Data". In: *Proceedings of the 7th International Conference on Algebraic Methodology and Software Technology (AMAST'98)*. Ed. by A. M. Haeberer. Vol. 1548. LNCS. Springer, 1999, pp. 74–90. DOI: 10.1007/3-540-49253-4_8.

[87] J. F. Groote and M. R. Mousavi. *Modeling and Analysis of Communicating Systems*. MIT Press, 2014. URL: http://mitpress.mit.edu/books/modeling-and-analysis-communicating-systems.

[88] J. F. Groote and T. A. C. Willemse. "Model-checking processes with data". In: *Sci. Comput. Program.* 56.3 (2005), pp. 251–273. DOI: 10.1016/j.scico.2004.08.002.

[89] A. Gruler, M. Leucker, and K. D. Scheidemann. "Modeling and Model Checking Software Product Lines". In: *Proceedings of the 10th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'08)*. Ed. by G. Barthe and F. de Boer. Vol. 5051. LNCS. Springer, 2008, pp. 113–131. DOI: 10.1007/978-3-540-68863-1_8.

[90] H. Hermanns, ed. *Interactive Markov Chains and the Quest for Quantified Quality*. Vol. 2428. LNCS. Springer, 2002. DOI: 10.1007/3-540-45804-2.

[91] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1995. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521673532.

[92] G. Iacobelli, M. Tribastone, and A. Vandin. "Differential Bisimulation for a Markovian Process Algebra". In: *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS'15)*. Ed. by G. F. Italiano, G. Pighizzini, and D. Sannella. Vol. 9234. LNCS. Springer, 2015, pp. 293–306. DOI: 10.1007/978-3-662-48057-1_23.

[93] J.-P. Katoen. "The Probabilistic Model Checking Landscape". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'16)*. ACM, 2016, pp. 31–45. DOI: 10.1145/2933575.2934574.

[94] C. H. P. Kim, D. Marinov, S. Khurshid, D. S. Batory, S. Souto, P. Barros, and M. d'Amorim. "SPLat: Lightweight Dynamic Analysis for Reducing Combinatorics in Testing Configurable Systems". In: *Proceedings of the Joint 14th European Software Engineering Conference and 21st International Symposium on Foundations of Software Engineering (ESEC/FSE'13)*. ACM, 2013, pp. 257–267. DOI: 10.1145/2491411.2491459.

[95] M. Kowal, I. Schaefer, and M. Tribastone. "Family-Based Performance Analysis of Variant-Rich Software Systems". In: *Proceedings of the 17th International Conference on Fundamental Approaches to Software Engineering (FASE'14)*. Vol. 8411. LNCS. Springer, 2014, pp. 94–108. DOI: 10.1007/978-3-642-54804-8_7.

[96] D. Kozen. "Results on the Propositional mu-Calculus". In: *Theoret. Comput. Sci.* 27 (1983), pp. 333–354. DOI: 10.1016/0304-3975(82)90125-6.

[97] K. G. Larsen and A. Legay. "Statistical Model Checking: Past, Present, and Future". In: *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*. 2016, pp. 3–15. DOI: 10.1007/978-3-319-47166-2_1.

[98] K. Larsen and A. Skou. "Bisimulation through probabilistic testing". In: *Information and Computation* 94.1 (1991), pp. 1–28. DOI: 10.1016/0890-5401(91)90030-6.

[99] D. Latella, M. Loreti, and M. Massink. "On-the-fly Probabilistic Model Checking". In: *Proceedings 7th Interaction and Concurrency Experience (ICE'14)*. Ed. by I. Lanese, A. L. Lafuente, A. Sokolova, and H. T. Vieira. Vol. 166. EPTCS. 2014, pp. 45–59. DOI: 10.4204/EPTCS.166.6.

[100] G. Li and H. Rabitz. "A general analysis of exact lumping in chemical kinetics". In: *Chemical Engineering Science* 44.6 (1989), pp. 1413–1430. DOI: 10.1016/0009-2509(89)85014-6.

[101] S. Lity, T. Morbach, T. Thüm, and I. Schaefer. "Applying Incremental Model Slicing to Product-Line Regression Testing". In: *Proceedings of the 15th International Conference on Software Reuse (ICSR'16)*. Ed. by G. M. Kapitsaki and E. S. de Almeida. Vol. 9679. LNCS. Springer, 2016, pp. 3–19. DOI: 10.1007/978-3-319-35122-3_1.

[102] M. Loreti and J. Hillston. "Modelling and Analysis of Collective Adaptive Systems with CARMA and its Tools". In: *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems - 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2016, Bertinoro, Italy, June 20-24, 2016, Advanced Lectures*. Ed. by M. Bernardo, R. D. Nicola, and J. Hillston. Vol. 9700. Lecture Notes in Computer Science. Springer, 2016, pp. 83–119. DOI: 10.1007/978-3-319-34096-8_4.

[103] O. Maler and D. Nickovic. "Monitoring Temporal Properties of Continuous Signals". In: *Proceedings of the Joint International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS'04) and Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'04)*. Ed. by Y. Lakhnech and S. Yovine. Vol. 3253. LNCS. Springer, 2004, pp. 152–166. DOI: 10.1007/978-3-540-30206-3_12.

[104] C. Maurer, R. Qi, and V. Raghavan. "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 25.2 (2003), pp. 265–270. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2003.1177156.

[105] L. M. de Moura and N. Bjørner. "Z3: An Efficient SMT Solver". In: *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*. Ed. by C. R. Ramakrishnan and J. Rehof. Vol. 4963. LNCS. Springer, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[106] L. Nenzi and L. Bortolussi. "Specifying and Monitoring Properties of Stochastic Spatio-Temporal Systems in Signal Temporal Logic". In: *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools (ValueTools'14)*. Ed. by M. Haviv, W. J. Knottenbelt, L. Maggi, and D. Miorandi. ICST, 2014. DOI: 10.4108/icst.valuetools.2014.258183.

[107] M. Okino and M. Mavrovouniotis. "Simplification of Mathematical Models of Chemical Reaction Systems". In: *Chemical Reviews* 2.98 (1998), pp. 391–408. DOI: 10.1021/cr9502231.

[108] R. Paige and R. Tarjan. "Three Partition Refinement Algorithms". In: *SIAM J. Comput.* 16.6 (1987), pp. 973–989. DOI: 10.1137/0216062.

[109] P. Piho. "Spatial and Stochastic Equivalence Relations for PALOMA". MA thesis. School of Informatics, University of Edinburgh, 2016.

[110] M. Plath and M. Ryan. "Feature integration using a feature construct". In: *Sci. Comput. Program.* 41.1 (2001), pp. 53–84. DOI: 10.1016/S0167-6423(00)00018-6.

[111] P. J. Ramadge and W. M. Wonham. "Supervisory control of a class of discrete event processes". In: *SIAM J. Control Optim.* 25.1 (1987), pp. 206–230. DOI: 10.1137/0325013.

[112] C. K. I. Rasmussen C. E. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[113] I. Schaefer. "Variability Modelling for Model-Driven Development of Software Product Lines". In: *Proceedings of the 4th International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS'10)*. 2010, pp. 85–92.

[114] S. Sebastio and A. Vandin. "MultiVeStA: Statistical Model Checking for Discrete Event Simulators". In: *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools (ValueTools'13)*. Ed. by A. Horvath, P. Buchholz, V. Cortellessa, L. Muscariello, and M. S. Squillante. ACM, 2013, pp. 310–315. DOI: 10.4108/icst.valuetools.2013.254377.

[115] M. B. Smyth and J. Webster. "Discrete Spatial Models". In: *Handbook of Spatial Logics*. Ed. by M. Aiello, I. E. Pratt-Hartmann, and J. F. A. K. van Benthem. Springer, 2007, pp. 713–798. DOI: 10.1007/978-1-4020-5587-4_12.

[116] W. J. Stewart. *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, 2009.

[117] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. "A Classification and Survey of Analysis Strategies for Software Product Lines". In: *ACM Comput. Surv.* 47.1 (2014), 6:1–6:45. DOI: 10.1145/2580950.

[118] T. Thüm, I. Schaefer, M. Hentschel, and S. Apel. "Family-Based Deductive Verification of Software Product Lines". In: *Proceedings of the 11th International Conference on Generative Programming and Component Engineering (GPCE'12)*. ACM, 2012, pp. 11–20. DOI: 10.1145/2371401.2371404.

[119] J. Toth, G. Li, H. Rabitz, and A. Tomlin. "The Effect of Lumping and Expanding on Kinetic Differential Equations". English. In: *SIAM J. Appl. Math.* 57.6 (1997), pp. 1531–1556. DOI: 10.1137/S0036139995293294.

[120] M. Tschaikowski and M. Tribastone. "A unified framework for differential aggregations in Markovian process algebra". In: *J. Log. Algebr. Meth. Program.* 84.2 (2015), pp. 238–258. DOI: 10.1016/j.jlamp.2014.10.004.

[121] T. Turanyi and A. S. Tomlin. *Analysis of Kinetic Reaction Mechanisms*. Springer, 2014. DOI: 10.1007/978-3-662-44562-4.

[122] A. Valmari and G. Franceschinis. "Simple $O(m \log n)$ Time Markov Chain Lumping". In: *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*. Vol. 6015. LNCS. Springer, 2010, pp. 38–52. DOI: 10.1007/978-3-642-12002-2_4.

[123] N. G. Van Kampen. *Stochastic processes in physics and chemistry*. Vol. 1. Elsevier, 1992.